FREE DEMO SAMPLE

# Guessing Paper 2026-27 Demo Set

## BCA · Semester 1 · Principles of Programming using C

SMKV Bastar — शहीद महेन्द्र कर्मा विश्वविद्यालय, बस्तर

### यह सिर्फ़ एक डेमो सेट है 📘

इस डेमो से आप हमारे उत्तरों की क्वालिटी और पेपर का फ़ॉर्मेट देख सकते हैं। **असली 3 पूरे Guessing Paper सेट** (Set A, B, C) सिर्फ़ **Pariksha Pass App** में मिलते हैं:

- हर विषय के **3 पूरे सेट** — Most Probable, Highly Likely और Alternate
- हर प्रश्न का **सटीक मॉडल उत्तर** — हिंदी और English दोनों में
- **Year Tag** — जो प्रश्न पहले यूनिवर्सिटी परीक्षा में आ चुके हैं
- आपकी **यूनिवर्सिटी के पैटर्न** पर — पूरा सिलेबस कवर

**▶ Pariksha Pass App डाउनलोड करें**
**यहाँ टैप करें — Google Play पर उपलब्ध**

QR स्कैन करें

इस डेमो के प्रश्न असली Set A/B/C से नहीं लिए गए हैं। कोई भी पेपर परीक्षा में प्रश्न आने की गारंटी नहीं देता — इसे पाठ्यपुस्तक के साथ अभ्यास के लिए उपयोग करें। · Free sample shared on cguniversities.com

**DEMO SAMPLE – NOT FROM THE ACTUAL SETS**
**डेमो नमूना – वास्तविक सेटों के प्रश्न इसमें नहीं हैं**

Paper Code: **6663**
Demo Set

Roll No. | | | | | | | | | |

# SHAHEED MAHENDRA KARMA VISHWAVIDYALAYA, BASTAR (C.G.)
## शहीद महेन्द्र कर्मा विश्वविद्यालय, बस्तर (छ.ग.)

**B.C.A. (First Semester) Examination, Nov.–Dec. 2026**
**बी.सी.ए. (प्रथम सेमेस्टर) परीक्षा, नवम्बर–दिसम्बर 2026**

# PRINCIPLES OF PROGRAMMING USING C
# C भाषा का उपयोग करते हुए प्रोग्रामिंग के सिद्धांत

| Time / समय : 3 Hours | Maximum Marks / पूर्णांक : 70 |
|---|---|

*DEMO SAMPLE – not from the actual sets · prepared by Online Student Seva to show the paper format and answer quality · not an official University question paper*

**Instructions:**

1. Section A: Q.1 (objective) is compulsory. All 5 short-answer questions in Q.2 are compulsory.
2. Section B: attempt **one** question from each unit (Q.3 to Q.6); each has an internal choice.
3. Figures in the right margin show full marks. Draw neat, labelled diagrams wherever necessary.

**निर्देश :** खण्ड-अ में प्रश्न 1 अनिवार्य है; प्रश्न 2 के सभी 5 प्रश्न अनिवार्य हैं। खण्ड-ब में प्रत्येक इकाई से **एक** प्रश्न हल करें (प्रश्न 3 से 6), सभी में आंतरिक विकल्प है। दाईं ओर के अंक पूर्णांक दर्शाते हैं। आवश्यकतानुसार स्वच्छ व नामांकित चित्र बनाएँ।

## SECTION – A / खण्ड – अ

**Q.1** **Choose the correct answer.** सही उत्तर चुनिए। 10 × 1 = 10

**(1)** What is the value of the expression 7 / 2 in C when both operands are integers?
C में जब दोनों संकार्य (ऑपरेंड) पूर्णांक हों तो व्यंजक 7 / 2 का मान क्या होगा?

(a) 3.5 / 3.5 (b) 2 / 2
(c) 4 / 4 (d) 3 / 3

**(2)** Which escape sequence is used in printf() to print a horizontal tab?
printf() में क्षैतिज टैब (horizontal tab) छापने के लिए कौन-सा एस्केप सीक्वेंस प्रयुक्त होता है?

(a) \t / \t (b) \n / \n
(c) \b / \b (d) \a / \a

**(3)** Which of the following is a logical operator in C?
निम्नलिखित में से कौन-सा C का तार्किक (लॉजिकल) ऑपरेटर है?

(a) & / & (b) ?: / ?:
(c) == / == (d) && / &&

**(4)** What is the output of printf("%d", 5 > 3 && 2 > 4); in C?
C में printf("%d", 5 > 3 && 2 > 4); का आउटपुट क्या है?

(a) 0 / 0 (b) 1 / 1
(c) 2 / 2 (d) Compilation error / कंपाइलेशन त्रुटि

© Online Student Seva · Exclusive property · Accessible only through the Pariksha Pass App · Copying, sharing or selling is strictly prohibited and will invite serious legal action.
© ऑनलाइन स्टूडेंट सेवा · विशेष संपत्ति · केवल परीक्षा पास ऐप पर उपलब्ध · कॉपी, शेयर या बिक्री पर कड़ी कानूनी कार्रवाई होगी।

**(5)** In a switch statement, the default case is:
switch स्टेटमेंट में default केस:

(a) Compulsory and must be the last / अनिवार्य और अंतिम होना चाहिए
(b) Optional / वैकल्पिक (ऑप्शनल) है
(c) Compulsory and must be the first / अनिवार्य और पहला होना चाहिए
(d) Allowed only with integer values / केवल पूर्णांक मानों के साथ मान्य है

**(6)** The library function strcmp(s1, s2) returns 0 when:
लाइब्रेरी फंक्शन strcmp(s1, s2) शून्य (0) तब लौटाता है जब:

(a) s1 is longer than s2 / s1, s2 से लंबी हो
(b) s1 is shorter than s2 / s1, s2 से छोटी हो
(c) both strings are equal / दोनों स्ट्रिंग समान हों
(d) s2 is empty / s2 खाली हो

**(7)** Which of the following is essential in a recursive function to avoid infinite recursion?
अनंत रिकर्शन से बचने के लिए रिकर्सिव फंक्शन में निम्न में से क्या आवश्यक है?

(a) A global variable / ग्लोबल वेरिएबल
(b) A base condition / आधार शर्त (बेस कंडीशन)
(c) A pointer argument / पॉइंटर आर्ग्युमेंट
(d) A return type of void / void रिटर्न टाइप

**(8)** The arrow operator (->) is used to access a member of a structure through:
तीर ऑपरेटर (->) का उपयोग स्ट्रक्चर के सदस्य तक किसके माध्यम से पहुँचने के लिए होता है?

(a) The structure variable name / स्ट्रक्चर वेरिएबल के नाम से
(b) A pointer to the structure / स्ट्रक्चर के पॉइंटर से
(c) An array of integers / पूर्णांकों की ऐरे से
(d) A union tag / यूनियन टैग से

**(9)** What does fopen() return if the file cannot be opened?
यदि फाइल खोली न जा सके तो fopen() क्या लौटाता है?

(a) 0 as an integer error code / पूर्णांक त्रुटि कोड 0
(b) -1 / -1
(c) NULL / NULL
(d) EOF / EOF

**(10)** A local variable declared with which keyword retains its value between successive calls of the function?
किस कीवर्ड से घोषित स्थानीय वेरिएबल फंक्शन की क्रमिक कॉलों के बीच अपना मान बनाए रखता है?

(a) auto / auto
(b) register / register
(c) static / static
(d) typedef / typedef

**Q.2 Answer the following questions.** निम्नलिखित प्रश्नों के उत्तर दीजिए। 5 × 4 = 20

**(a)** What are escape sequences in C? List any five with their meaning and give an example. 4
C में एस्केप सीक्वेंस क्या हैं? कोई पाँच अर्थ सहित लिखिए और एक उदाहरण दीजिए।

**(b)** Write a C program to print the multiplication table of a number using a for loop. 4
for लूप का प्रयोग करके किसी संख्या की गुणन-सारणी (पहाड़ा) छापने का C प्रोग्राम लिखिए।

**(c)** What is a function prototype in C? Why is it needed? Explain with an example. 4
C में फंक्शन प्रोटोटाइप क्या है? यह क्यों आवश्यक है? उदाहरण सहित समझाइए।

**(d)** Explain the void pointer and the NULL pointer in C with examples. 4
C में void पॉइंटर और NULL पॉइंटर को उदाहरण सहित समझाइए।

© Online Student Seva · Exclusive property · Accessible only through the Pariksha Pass App · Copying, sharing or selling is strictly prohibited and will invite serious legal action.
© ऑनलाइन स्टूडेंट सेवा · विशेष संपत्ति · केवल परीक्षा पास ऐप पर उपलब्ध · कॉपी, शेयर या बिक्री पर कड़ी कानूनी कार्रवाई होगी।

**(e)** Differentiate between text files and binary files in C. 4
C में टेक्स्ट फाइल तथा बाइनरी फाइल में अंतर कीजिए।

## SECTION – B / खण्ड – ब

Long answer questions · one question from each unit with internal choice / **दीर्घ उत्तरीय प्रश्न**

**Q.3** What are unformatted input and output functions in C? Explain getchar(), putchar(), gets(), puts() and fgets() with programs, and distinguish them from scanf() and printf(). 10
C में अनफॉर्मेटेड इनपुट और आउटपुट फंक्शन क्या हैं? getchar(), putchar(), gets(), puts() तथा fgets() को प्रोग्राम सहित समझाइए और इन्हें scanf() व printf() से अलग कीजिए।

**OR / अथवा**

What are bitwise operators in C? Explain all six bitwise operators with truth tables and a program, and describe their applications such as testing odd or even, swapping and setting or clearing a bit.
C में बिटवाइज़ ऑपरेटर क्या हैं? सभी छह बिटवाइज़ ऑपरेटरों को सत्यता सारणी और प्रोग्राम सहित समझाइए तथा विषम या सम की जाँच, अदला-बदली और बिट को सेट या क्लियर करने जैसे उनके अनुप्रयोग बताइए।

**Q.4** Write C programs (i) to convert a decimal number into binary, (ii) to find the GCD and LCM of two numbers using Euclid's method, and (iii) to print the perfect numbers up to 500. Explain the logic of each. 10
C प्रोग्राम लिखिए (i) दशमलव संख्या को बाइनरी में बदलने का, (ii) यूक्लिड की विधि से दो संख्याओं का महत्तम समापवर्तक (GCD) और लघुत्तम समापवर्त्य (LCM) निकालने का, तथा (iii) 500 तक की पूर्ण संख्याएँ छापने का। प्रत्येक का तर्क समझाइए।

**OR / अथवा**

Write C programs (i) to find the roots of a quadratic equation using the if-else-if ladder, (ii) to find the largest of three numbers using nested if and the conditional operator, and (iii) to check whether a character is a vowel or a consonant using switch. Explain each.
C प्रोग्राम लिखिए (i) if-else-if सीढ़ी से द्विघात समीकरण के मूल निकालने का, (ii) नेस्टेड if और कंडीशनल ऑपरेटर से तीन संख्याओं में सबसे बड़ी संख्या निकालने का, तथा (iii) switch से जाँचने का कि कोई वर्ण स्वर है या व्यंजन। प्रत्येक को समझाइए।

**Q.5** What is searching? Explain linear search and binary search with algorithms, programs and outputs. Compare the two methods. 10
खोज (सर्चिंग) क्या है? रैखिक खोज (लीनियर सर्च) और द्विआधारी खोज (बाइनरी सर्च) को एल्गोरिदम, प्रोग्राम और आउटपुट सहित समझाइए। दोनों विधियों की तुलना कीजिए।

**OR / अथवा**

How are strings stored in C? Write user-defined functions equivalent to strlen(), strcpy(), strcat() and strcmp() without using the string library, and demonstrate them with a program.
C में स्ट्रिंग कैसे संग्रहीत होती हैं? स्ट्रिंग लाइब्रेरी का उपयोग किए बिना strlen(), strcpy(), strcat() तथा strcmp() के समतुल्य यूज़र-डिफ़ाइंड फंक्शन लिखिए और प्रोग्राम से उन्हें प्रदर्शित कीजिए।

© Online Student Seva · Exclusive property · Accessible only through the Pariksha Pass App · Copying, sharing or selling is strictly prohibited and will invite serious legal action.
© ऑनलाइन स्टूडेंट सेवा · विशेष संपत्ति · केवल परीक्षा पास ऐप पर उपलब्ध · कॉपी, शेयर या बिक्री पर कड़ी कानूनी कार्रवाई होगी।

**Q.6** What is an array of structures? Explain how a pointer to a structure is used to pass a structure to a function. Write a program that stores the marks of three students, calculates the total and average and finds the topper. **10**

संरचनाओं की ऐरे (ऐरे ऑफ स्ट्रक्चर) क्या है? फंक्शन को स्ट्रक्चर भेजने के लिए स्ट्रक्चर के पॉइंटर का उपयोग कैसे होता है, समझाइए। ऐसा प्रोग्राम लिखिए जो तीन विद्यार्थियों के अंक संग्रहीत करे, कुल और औसत निकाले तथा टॉपर ज्ञात करे।

**OR / अथवा**

What is random access in files? Explain fseek(), ftell(), rewind() and the constants SEEK_SET, SEEK_CUR and SEEK_END. Write a program that stores records in a binary file and updates the second record in place.

फाइलों में रैंडम एक्सेस क्या है? fseek(), ftell(), rewind() तथा स्थिरांक SEEK_SET, SEEK_CUR और SEEK_END को समझाइए। ऐसा प्रोग्राम लिखिए जो रिकॉर्डों को बाइनरी फाइल में संग्रहीत करे और दूसरे रिकॉर्ड को उसी स्थान पर अद्यतन (अपडेट) करे।

**— END OF QUESTION PAPER · प्रश्न-पत्र समाप्त —**

DEMO SAMPLE · ONLINE STUDENT SEVA

© Online Student Seva · Exclusive property · Accessible only through the Pariksha Pass App · Copying, sharing or selling is strictly prohibited and will invite serious legal action.
© ऑनलाइन स्टूडेंट सेवा · विशेष संपत्ति · केवल परीक्षा पास ऐप पर उपलब्ध · कॉपी, शेयर या बिक्री पर कड़ी कानूनी कार्रवाई होगी।

## Syllabus Coverage & Reference Books पाठ्यक्रम कवरेज एवं संदर्भ ग्रंथ

This paper follows the official **Principles of Programming using C** syllabus (6663) of the Chhattisgarh Higher Education Department (NEP-2020). The table shows what each unit contains and where this paper tests it.

| Unit | Syllabus content (as prescribed) | Questions in this set |
|---|---|---|
| 1 | **Fundamentals of C: Variables, Data Types and Operators**<br>Structure of a C program, compilation process, comments, keywords and identifiers, variables and constants, data types and type conversion, input and output functions, all operators with precedence, preprocessor directives and macros.<br>**C के मूल तत्व: वेरिएबल, डेटा टाइप और ऑपरेटर** — C प्रोग्राम की संरचना, कंपाइलेशन प्रक्रिया, टिप्पणियाँ (कमेंट), कीवर्ड और आइडेंटिफायर, वेरिएबल और कांस्टेंट, डेटा टाइप और टाइप कन्वर्शन, इनपुट-आउटपुट फंक्शन, सभी ऑपरेटर और उनकी प्राथमिकता, प्रीप्रोसेसर निर्देश और मैक्रो। | Q.1(1), Q.1(2), Q.1(3), Q.2(a), Q.3 |
| 2 | **Control Statements and Loops**<br>if, if-else, else-if ladder, nested if, switch statement and its difference from if-else, for, while and do-while loops, break, continue and goto, nested loops, and programs such as the calculator, series and pattern printing.<br>**कंट्रोल स्टेटमेंट और लूप** — if, if-else, else-if सीढ़ी, नेस्टेड if, switch स्टेटमेंट और if-else से उसका अंतर, for, while और do-while लूप, break, continue और goto, नेस्टेड लूप तथा कैलकुलेटर, श्रेणी और पैटर्न प्रिंटिंग जैसे प्रोग्राम। | Q.1(4), Q.1(5), Q.2(b), Q.4 |
| 3 | **Functions, Arrays, Strings and Recursion**<br>Function declaration, definition and call, types of functions, call by value and call by reference, recursion, one- and two-dimensional arrays, passing arrays to functions, strings and string library functions, sorting and matrix programs.<br>**फंक्शन, ऐरे, स्ट्रिंग और रिकर्शन** — फंक्शन डिक्लेरेशन, डेफिनिशन और कॉल, फंक्शन के प्रकार, कॉल बाय वैल्यू और कॉल बाय रेफरेंस, रिकर्शन, एक और द्वि-आयामी ऐरे, फंक्शन को ऐरे भेजना, स्ट्रिंग और स्ट्रिंग लाइब्रेरी फंक्शन, सॉर्टिंग और मैट्रिक्स प्रोग्राम। | Q.1(6), Q.1(7), Q.2(c), Q.5 |
| 4 | **Number Programs, Pointers, Structures and Files**<br>Standard programs (prime, factorial, minimum and maximum, reversing digits, positive or negative, Armstrong, leap year), pointers and pointer arithmetic, structures and unions, storage classes, dynamic memory allocation and file handling.<br>**संख्या प्रोग्राम, पॉइंटर, स्ट्रक्चर और फाइलें** — मानक प्रोग्राम (अभाज्य संख्या, फैक्टोरियल, न्यूनतम और अधिकतम, अंकों को उलटना, धनात्मक या ऋणात्मक, आर्मस्ट्रॉन्ग, लीप वर्ष), पॉइंटर और पॉइंटर अंकगणित, स्ट्रक्चर और यूनियन, स्टोरेज क्लास, डायनामिक मेमोरी आवंटन और फाइल हैंडलिंग। | Q.1(8), Q.1(9), Q.1(10), Q.2(d), Q.2(e), Q.6 |

### Books recommended in the syllabus (derive your answers from these) / पाठ्यक्रम में अनुशंसित पुस्तकें

**Text books:**

- E. Balagurusamy - Programming in ANSI C; Tata McGraw-Hill (TMH)
- Yashavant Kanetkar - Let Us C; BPB Publications
- Brian W. Kernighan and Dennis M. Ritchie - The C Programming Language; Prentice Hall of India (PHI)
- Byron S. Gottfried - Programming with C (Schaum's Outline Series); McGraw-Hill

© Online Student Seva · Exclusive property · Accessible only through the Pariksha Pass App · Copying, sharing or selling is strictly prohibited and will invite serious legal action.
© ऑनलाइन स्टूडेंट सेवा · विशेष संपत्ति · केवल परीक्षा पास ऐप पर उपलब्ध · कॉपी, शेयर या बिक्री पर कड़ी कानूनी कार्रवाई होगी।

**Reference books:**

- Herbert Schildt - C: The Complete Reference; McGraw-Hill
- Reema Thareja - Programming in C; Oxford University Press
- Ashok N. Kamthane - Programming in C; Pearson Education
- Stephen G. Kochan - Programming in C; Addison-Wesley

Always use the latest edition. Model answers in this paper follow the standard university textbooks listed above, in simplified exam-ready form. / नवीनतम संस्करण का प्रयोग करें।

**Important advice / ज़रूरी सलाह**

This paper is based on the NEP syllabus and high-probability topic coverage, and it covers the topics most likely to be asked. However, **do not depend on this paper alone** — prepare the complete syllabus from your text book. Use this paper to practise writing and time management.

यह पेपर एनईपी पाठ्यक्रम और अधिक संभावित टॉपिक कवरेज पर आधारित है तथा अधिक संभावित टॉपिक कवर करता है। फिर भी **केवल इसी पर निर्भर न रहें** — अपनी पाठ्यपुस्तक से पूरा पाठ्यक्रम तैयार करें। इस पेपर का उपयोग लेखन-अभ्यास व समय-प्रबंधन के लिए करें।

DEMO SAMPLE · ONLINE STUDENT SEVA

© Online Student Seva · Exclusive property · Accessible only through the Pariksha Pass App · Copying, sharing or selling is strictly prohibited and will invite serious legal action.
© ऑनलाइन स्टूडेंट सेवा · विशेष संपत्ति · केवल परीक्षा पास ऐप पर उपलब्ध · कॉपी, शेयर या बिक्री पर कड़ी कानूनी कार्रवाई होगी।

# PART – I : English Medium – Model Answers

Every answer is written to the marks allotted — learn the structure, then write in your own words.

## Section A · Q.1 : Answer key

| No. | Ans. | Correct option | Explanation |
|---|---|---|---|
| 1 | (d) | 3 | Division of two integers gives an integer result, and the fractional part is discarded, so 7 / 2 = 3. To get 3.5 one operand must be float, e.g. 7 / 2.0. |
| 2 | (a) | \t | \t prints a horizontal tab. \n is a new line, \b a backspace and \a an audible alert. |
| 3 | (d) | && | && is the logical AND operator. & is a bitwise operator, == is a relational operator and ?: is the conditional operator. |
| 4 | (a) | 0 | 5 > 3 is true (1) but 2 > 4 is false (0). The expression 1 && 0 is false, and a false relational or logical result prints as 0. |
| 5 | (b) | Optional | The default case is optional. It runs when no case label matches the switch expression. |
| 6 | (c) | both strings are equal | strcmp() returns 0 if both strings are identical, a negative value if s1 is less than s2, and a positive value if s1 is greater than s2. |
| 7 | (b) | A base condition | A base condition stops the function from calling itself again. Without it the calls continue until the stack overflows. |
| 8 | (b) | A pointer to the structure | The dot operator is used with a structure variable, while the arrow operator is used with a pointer to a structure: ptr->member is the same as (*ptr).member. |
| 9 | (c) | NULL | fopen() returns a FILE pointer, and it returns NULL on failure. Programs should always check this value before reading or writing. |
| 10 | (c) | static | A static local variable is created once and keeps its value until the program ends. An auto variable is re-created on every call. |

## Section A · Q.2 : Short answers

**Q.2(a)** 4 marks · ≈ 100–120 words

**What are escape sequences in C? List any five with their meaning and give an example.**

Escape sequences are special character combinations that begin with a backslash. They represent characters, such as a new line, that cannot be typed directly inside a string. The compiler treats each sequence as one character.

| Sequence | Meaning |
|---|---|
| \n | New line |
| \t | Horizontal tab |
| \b | Backspace |
| \\ | Backslash |
| \" | Double quote |

**Example:**

© Online Student Seva · Exclusive property · Accessible only through the Pariksha Pass App · Copying, sharing or selling is strictly prohibited and will invite serious legal action.
© ऑनलाइन स्टूडेंट सेवा · विशेष संपत्ति · केवल परीक्षा पास ऐप पर उपलब्ध · कॉपी, शेयर या बिक्री पर कड़ी कानूनी कार्रवाई होगी।

```
printf("Name:\tAnil\n\"C\" is fun\n");
```

**Output (tab shown as spaces):**

```
Name:    Anil
"C" is fun
```

Here \t moves to the next tab stop, \n starts a new line, and \" prints the quotation marks without ending the string. Other useful sequences are \a for an alert and \0 for the null character that ends a string.

**Q.2(b)** 4 marks · ≈ 100–120 words

**Write a C program to print the multiplication table of a number using a for loop.**

**Logic:** keep the number in n and repeat the multiplication n x i for i = 1 to 10, printing one line in each pass of the loop.

```
#include <stdio.h>

int main(void)
{
    int n = 7, i;
    for (i = 1; i <= 10; i++)
        printf("%d x %2d = %2d\n", n, i, n * i);
    return 0;
}
```

**Output (first and last lines):**

```
7 x  1 =  7
7 x  2 = 14
...
7 x 10 = 70
```

The loop variable i starts at 1, the condition i <= 10 is tested before each pass, and i++ increases it after each pass, so the body runs exactly 10 times. The width %2d keeps the columns aligned.

**Q.2(c)** 4 marks · ≈ 100–120 words

**What is a function prototype in C? Why is it needed? Explain with an example.**

A function prototype (declaration) tells the compiler the name, return type and parameter types of a function before the function is used. Its form is: return_type name(type1, type2);

C reads a program from top to bottom. If main() calls a function that is defined below it, the compiler must already know how to check the call. Without a prototype, older compilers assume int, and newer ones give a warning or an error.

© Online Student Seva · Exclusive property · Accessible only through the Pariksha Pass App · Copying, sharing or selling is strictly prohibited and will invite serious legal action.
© ऑनलाइन स्टूडेंट सेवा · विशेष संपत्ति · केवल परीक्षा पास ऐप पर उपलब्ध · कॉपी, शेयर या बिक्री पर कड़ी कानूनी कार्रवाई होगी।

```
int add(int, int);                  /* prototype */

int main(void)
{
    printf("%d", add(2, 3));        /* prints 5 */
    return 0;
}

int add(int a, int b)               /* definition */
{
    return a + b;
}
```

A prototype allows type checking of arguments and lets us define functions after main().

**Q.2(d)** 4 marks · ≈ 100–120 words

**Explain the void pointer and the NULL pointer in C with examples.**

**Void pointer.** A void pointer (void *p) is a generic pointer that can hold the address of a variable of any data type. It cannot be dereferenced directly, because the type and size of the data are unknown; it must first be type-cast.

```
int x = 10;
void *p = &x;
printf("%d", *(int *)p);    /* prints 10 */
```

**NULL pointer.** A NULL pointer points to nothing. It is written p = NULL; where NULL is a macro defined in stdio.h and stddef.h with the value 0. Dereferencing it is an error, so a pointer is tested with if (p != NULL) before use. It is used to initialise pointers and to mark the end of a linked list.

**Q.2(e)** 4 marks · ≈ 100–120 words

**Differentiate between text files and binary files in C.**

A text file stores data as readable characters, while a binary file stores the bytes exactly as they are in memory.

| Point | Text file | Binary file |
|---|---|---|
| Content | Readable characters | Raw bytes, not readable |
| Modes | "r", "w", "a" | "rb", "wb", "ab" |
| Functions | fprintf, fscanf, fgets, fputs | fread, fwrite |
| Newline | Translated, e.g. \n to \r\n on Windows | Not translated |
| Size | Larger for numbers | Compact |
| Speed | Slower, conversion needed | Faster, no conversion |

For example, the integer 12345 takes 5 bytes (one per digit) in a text file but only 4 bytes as a binary int on a typical system. Binary files are preferred for records of structures, and text files for reports and configuration.

© Online Student Seva · Exclusive property · Accessible only through the Pariksha Pass App · Copying, sharing or selling is strictly prohibited and will invite serious legal action.
© ऑनलाइन स्टूडेंट सेवा · विशेष संपत्ति · केवल परीक्षा पास ऐप पर उपलब्ध · कॉपी, शेयर या बिक्री पर कड़ी कानूनी कार्रवाई होगी।

## Section B : Long answers

**Q.3 · Unit 1** 10 marks · ≈ 600–700 words

**What are unformatted input and output functions in C? Explain getchar(), putchar(), gets(), puts() and fgets() with programs, and distinguish them from scanf() and printf().**

### INTRODUCTION

Input and output in C are done through library functions declared in the header file stdio.h. They are of two kinds. The **formatted** functions, printf() and scanf(), convert data according to format specifiers such as %d and %f. The **unformatted** functions read or write raw characters or strings with no conversion. They are simpler and faster, and are useful for processing text one character or one line at a time.

### CHARACTER INPUT AND OUTPUT

| Function | Header | Purpose |
|---|---|---|
| getchar() | stdio.h | Reads one character from the keyboard (waits for Enter) |
| putchar(c) | stdio.h | Writes one character on the screen |
| getch() | conio.h | Reads one character without echo and without Enter |
| getche() | conio.h | Reads one character with echo and without Enter |

getchar() returns the character as an int, so that the special value EOF (end of file) can also be represented. The functions getch() and getche() belong to conio.h, which is not part of standard C: it is available in Turbo C and in Windows compilers, but not in gcc on Linux.

### STRING INPUT AND OUTPUT

- **puts(s)** writes the string s and automatically adds a new line.
- **gets(s)** reads a line into s, but it never checks the size of the array. This can overflow the buffer, so gets() was **removed from the C11 standard** and must not be used.
- **fgets(s, n, stdin)** is the safe replacement. It reads at most n - 1 characters, stops at the new line, keeps the new line in the string, and adds the null character.
- **scanf("%s", s)** stops at the first space, so it cannot read a full name.

DEMO SAMPLE · ONLINE STUDENT SEVA

© Online Student Seva · Exclusive property · Accessible only through the Pariksha Pass App · Copying, sharing or selling is strictly prohibited and will invite serious legal action.
© ऑनलाइन स्टूडेंट सेवा · विशेष संपत्ति · केवल परीक्षा पास ऐप पर उपलब्ध · कॉपी, शेयर या बिक्री पर कड़ी कानूनी कार्रवाई होगी।

### PROGRAM 1: CHARACTER PROCESSING WITH GETCHAR() AND PUTCHAR()

```
#include <stdio.h>
#include <ctype.h>

int main(void)
{
    int ch, letters = 0, digits = 0, others = 0;
    printf("Type a line: ");
    while ((ch = getchar()) != '\n' && ch != EOF) {
        if (isalpha(ch))
            letters++;
        else if (isdigit(ch))
            digits++;
        else
            others++;
        putchar(toupper(ch));
    }
    putchar('\n');
    printf("Letters=%d Digits=%d Others=%d\n", letters, digits, others);
    return 0;
}
```

**Output (gcc; input Raipur 492001!):**

```
Type a line: RAIPUR 492001!
Letters=6 Digits=6 Others=2
```

The loop reads one character at a time until the new line, classifies it, and prints its upper-case form. The functions isalpha() and isdigit() come from ctype.h and return a non-zero value when the character is a letter or a digit. The function toupper() converts a lower-case letter to upper case and leaves every other character unchanged.

### PROGRAM 2: READING A FULL LINE WITH FGETS() AND PUTS()

```
#include <stdio.h>
#include <string.h>

int main(void)
{
    char name[30];
    printf("Enter your full name: ");
    fgets(name, sizeof(name), stdin);
    name[strcspn(name, "\n")] = '\0';
    puts("Hello,");
    puts(name);
    printf("Length = %d\n", (int)strlen(name));
    return 0;
}
```

**Output (gcc; input Anil Kumar Verma):**

```
Enter your full name: Hello,
Anil Kumar Verma
Length = 16
```

The call strcspn(name, "\n") finds the position of the new line, which is replaced by the null character, so the length does not count it.

© Online Student Seva · Exclusive property · Accessible only through the Pariksha Pass App · Copying, sharing or selling is strictly prohibited and will invite serious legal action.
© ऑनलाइन स्टूडेंट सेवा · विशेष संपत्ति · केवल परीक्षा पास ऐप पर उपलब्ध · कॉपी, शेयर या बिक्री पर कड़ी कानूनी कार्रवाई होगी।

### DIFFERENCES BETWEEN UNFORMATTED AND FORMATTED FUNCTIONS

| Point | Unformatted (getchar, puts, fgets) | Formatted (scanf, printf) |
|---|---|---|
| Conversion | None, raw characters | Uses format specifiers |
| Data handled | Characters and strings | All data types |
| Spaces in input | fgets reads them | scanf stops at them |
| Speed | Faster | Slower |
| Use | Text processing | Numeric and mixed input |

### A COMMON PITFALL

After scanf("%d", &n) the new line typed by the user stays in the input buffer. A following getchar() or fgets() will read that new line at once and seem to be skipped. It is solved by consuming it with an extra getchar() call.

### CONCLUSION

Unformatted functions give simple, fast control over characters and lines. getchar() and putchar() suit character loops, puts() and fgets() suit lines, and gets() should be replaced by fgets() because it is unsafe.

**Q.3 (OR) · Unit 1** 10 marks · ≈ 600–700 words

**What are bitwise operators in C? Explain all six bitwise operators with truth tables and a program, and describe their applications such as testing odd or even, swapping and setting or clearing a bit.**

### INTRODUCTION

Bitwise operators work directly on the individual bits of integer operands (char, short, int, long). They cannot be applied to float or double. Because they act on the binary representation, they are fast and are widely used in device drivers, embedded programs, flags, compression and encryption.

### THE SIX BITWISE OPERATORS

| Operator | Name | Rule |
|---|---|---|
| & | AND | Result bit is 1 only if both bits are 1 |
| vertical bar | OR | Result bit is 1 if at least one bit is 1 |
| ^ | XOR | Result bit is 1 if the bits are different |
| ~ | NOT (complement) | Reverses every bit (unary) |
| << | Left shift | Moves bits left; each shift multiplies by 2 |
| >> | Right shift | Moves bits right; each shift divides by 2 |

**Truth table**

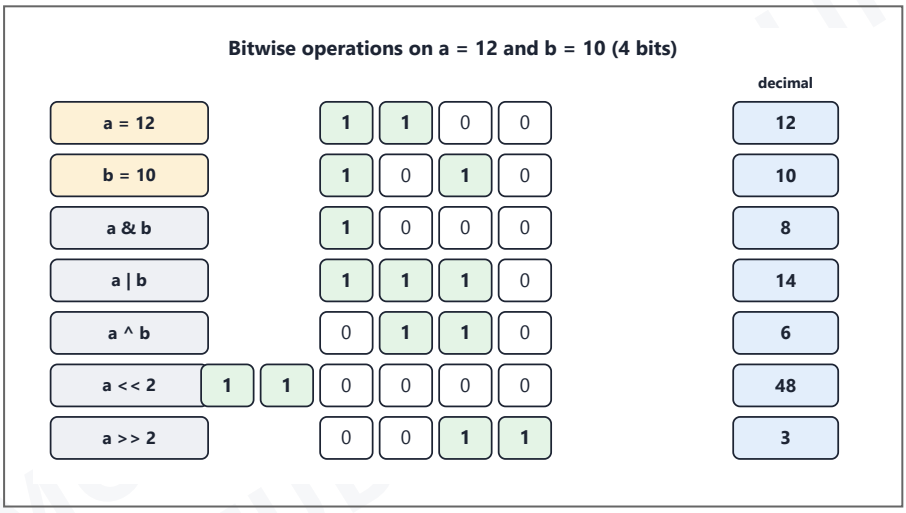
© Online Student Seva · Exclusive property · Accessible only through the Pariksha Pass App · Copying, sharing or selling is strictly prohibited and will invite serious legal action.
© ऑनलाइन स्टूडेंट सेवा · विशेष संपत्ति · केवल परीक्षा पास ऐप पर उपलब्ध · कॉपी, शेयर या बिक्री पर कड़ी कानूनी कार्रवाई होगी।

| a | b | a AND b | a OR b | a XOR b |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 | 0 |

## WORKED EXAMPLE

Let a = 12 (1100) and b = 10 (1010) in 4 bits.

- a & b = 1000 = 8
- a | b = 1110 = 14
- a ^ b = 0110 = 6
- ~a reverses all bits. On a machine using two's complement, ~n = -(n + 1), so ~12 = -13.
- a << 2 = 110000 = 48, which is 12 x $2^2$.
- a >> 2 = 0011 = 3, which is 12 / $2^2$ (for non-negative numbers).




*Bitwise operations on a = 12 and b = 10*

## PROGRAM 1: ALL OPERATORS

```
#include <stdio.h>

int main(void)
{
    int a = 12, b = 10;              /* 1100 and 1010 */
    printf("a & b  = %d\n", a & b);
    printf("a | b  = %d\n", a | b);
    printf("a ^ b  = %d\n", a ^ b);
    printf("~a     = %d\n", ~a);
    printf("a << 2 = %d\n", a << 2);
    printf("a >> 2 = %d\n", a >> 2);
    return 0;
}
```

**Output (gcc):**

© Online Student Seva · Exclusive property · Accessible only through the Pariksha Pass App · Copying, sharing or selling is strictly prohibited and will invite serious legal action.
© ऑनलाइन स्टूडेंट सेवा · विशेष संपत्ति · केवल परीक्षा पास ऐप पर उपलब्ध · कॉपी, शेयर या बिक्री पर कड़ी कानूनी कार्रवाई होगी।

```
a & b  = 8
a | b  = 14
a ^ b  = 6
~a     = -13
a << 2 = 48
a >> 2 = 3
```

## APPLICATIONS

1. **Odd or even:** the last bit of an odd number is 1, so n & 1 is non-zero for odd numbers.
2. **Swapping without a third variable:** a = a ^ b; b = a ^ b; a = a ^ b; because x ^ x = 0 and x ^ 0 = x.
3. **Power of two:** a positive n is a power of two if n & (n - 1) is 0, since only one bit is set.
4. **Set a bit:** x |= (1 << k).
5. **Clear a bit:** x &= ~(1 << k).
6. **Toggle a bit:** x ^= (1 << k).

## PROGRAM 2: APPLICATIONS

```
#include <stdio.h>

int main(void)
{
    int x = 25, p = 5, q = 9, n = 64;
    unsigned char r = 0;

    printf("%d is %s\n", x, (x & 1) ? "odd" : "even");
    p = p ^ q;
    q = p ^ q;
    p = p ^ q;
    printf("After swap: p = %d, q = %d\n", p, q);
    printf("%d %s a power of two\n", n,
           (n > 0 && (n & (n - 1)) == 0) ? "is" : "is not");
    r |= (1 << 3);                  /* set bit 3    */
    printf("After set    : %d\n", r);
    r ^= (1 << 0);                  /* toggle bit 0 */
    printf("After toggle : %d\n", r);
    r &= ~(1 << 3);                 /* clear bit 3  */
    printf("After clear  : %d\n", r);
    return 0;
}
```

**Output (gcc):**

```
25 is odd
After swap: p = 9, q = 5
64 is a power of two
After set    : 8
After toggle : 9
After clear  : 1
```

## POINTS TO REMEMBER

- & and | are bitwise, while && and || are logical operators; they must not be confused.
- Right-shifting a negative signed number is implementation-defined.

© Online Student Seva · Exclusive property · Accessible only through the Pariksha Pass App · Copying, sharing or selling is strictly prohibited and will invite serious legal action.
© ऑनलाइन स्टूडेंट सेवा · विशेष संपत्ति · केवल परीक्षा पास ऐप पर उपलब्ध · कॉपी, शेयर या बिक्री पर कड़ी कानूनी कार्रवाई होगी।

### CONCLUSION

Bitwise operators manipulate data at bit level. With the masks 1 << k they allow setting, clearing and testing of individual bits, and they give compact solutions to problems such as swapping and power-of-two tests.

**Q.4 · Unit 2** 10 marks · ≈ 600–700 words

**Write C programs (i) to convert a decimal number into binary, (ii) to find the GCD and LCM of two numbers using Euclid's method, and (iii) to print the perfect numbers up to 500. Explain the logic of each.**

### (I) DECIMAL TO BINARY

#### Logic

Divide the number repeatedly by 2 and store each remainder. The remainders, read in the reverse order (last one first), form the binary number. A loop is used because the number of divisions is not known in advance.

#### Program

```
#include <stdio.h>

int main(void)
{
    int n, bin[32], i = 0;
    printf("Enter a decimal number: ");
    scanf("%d", &n);
    printf("Binary of %d = ", n);
    if (n == 0)
        printf("0");
    while (n > 0) {
        bin[i] = n % 2;
        n = n / 2;
        i++;
    }
    for (i = i - 1; i >= 0; i--)
        printf("%d", bin[i]);
    printf("\n");
    return 0;
}
```

**Output (gcc; inputs 45 and 10):**

```
Enter a decimal number: Binary of 45 = 101101
Enter a decimal number: Binary of 10 = 1010
```

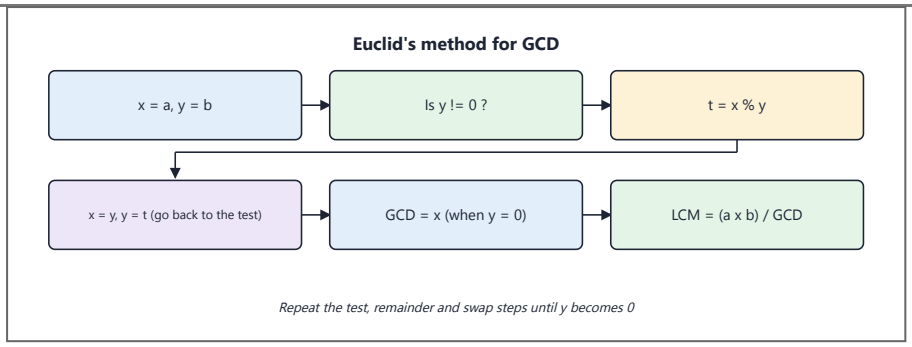
Dry run for 45: remainders 1, 0, 1, 1, 0, 1 are obtained in this order, and printing them backwards gives 101101.

### (II) GCD AND LCM BY EUCLID'S METHOD

#### Logic

The GCD of x and y is the same as the GCD of y and x % y. The step is repeated until the remainder becomes 0, and the last non-zero number is the GCD. The LCM is then found from the relation LCM = (a x b) / GCD.

© Online Student Seva · Exclusive property · Accessible only through the Pariksha Pass App · Copying, sharing or selling is strictly prohibited and will invite serious legal action.
© ऑनलाइन स्टूडेंट सेवा · विशेष संपत्ति · केवल परीक्षा पास ऐप पर उपलब्ध · कॉपी, शेयर या बिक्री पर कड़ी कानूनी कार्रवाई होगी।




*Flowchart of Euclid's method for GCD*

### Program

```
#include <stdio.h>

int main(void)
{
    int a, b, x, y, t;
    printf("Enter two numbers: ");
    scanf("%d %d", &a, &b);
    x = a;
    y = b;
    while (y != 0) {
        t = x % y;
        x = y;
        y = t;
    }
    printf("GCD = %d\n", x);
    printf("LCM = %d\n", (a * b) / x);
    return 0;
}
```

**Output (gcc; inputs 48 36 and 17 5):**

```
Enter two numbers: GCD = 12
LCM = 144
Enter two numbers: GCD = 1
LCM = 85
```

Dry run for 48 and 36: (x, y) = (48, 36), then (36, 12), then (12, 0). The loop stops and the GCD is 12, so LCM = 48 x 36 / 12 = 144.

## (III) PERFECT NUMBERS UP TO 500

### Logic

A number is perfect if it is equal to the sum of its proper divisors (divisors other than the number itself). For example, 6 = 1 + 2 + 3 and 28 = 1 + 2 + 4 + 7 + 14. An outer loop takes each number n, and an inner loop adds all divisors from 1 to n/2, because no proper divisor can exceed n/2.

© Online Student Seva · Exclusive property · Accessible only through the Pariksha Pass App · Copying, sharing or selling is strictly prohibited and will invite serious legal action.
© ऑनलाइन स्टूडेंट सेवा · विशेष संपत्ति · केवल परीक्षा पास ऐप पर उपलब्ध · कॉपी, शेयर या बिक्री पर कड़ी कानूनी कार्रवाई होगी।

### Program

```
#include <stdio.h>

int main(void)
{
    int n, i, sum;
    printf("Perfect numbers up to 500: ");
    for (n = 2; n <= 500; n++) {
        sum = 0;
        for (i = 1; i <= n / 2; i++) {
            if (n % i == 0)
                sum = sum + i;
        }
        if (sum == n)
            printf("%d ", n);
    }
    printf("\n");
    return 0;
}
```

**Output (gcc):**

```
Perfect numbers up to 500: 6 28 496
```

## NOTES

1. The array bin[32] is large enough because an int has at most 32 bits.
2. In the GCD program, a and b are copied into x and y so that the original values remain available for the LCM formula.
3. The product a x b can overflow for large inputs, so for big numbers the LCM is better calculated as (a / gcd) x b.

## SUMMARY OF THE THREE PROGRAMS

| Program | Main idea | Loop used |
|---|---|---|
| Decimal to binary | Repeated division by 2 | while, then for (reverse) |
| GCD and LCM | gcd(x, y) = gcd(y, x % y) | while |
| Perfect numbers | Sum of proper divisors | Nested for |

## CONCLUSION

All three programs depend on the % and / operators inside loops. Showing the dry run and the output proves the logic, and Euclid's method is far faster than checking every divisor.

© Online Student Seva · Exclusive property · Accessible only through the Pariksha Pass App · Copying, sharing or selling is strictly prohibited and will invite serious legal action.
© ऑनलाइन स्टूडेंट सेवा · विशेष संपत्ति · केवल परीक्षा पास ऐप पर उपलब्ध · कॉपी, शेयर या बिक्री पर कड़ी कानूनी कार्रवाई होगी।

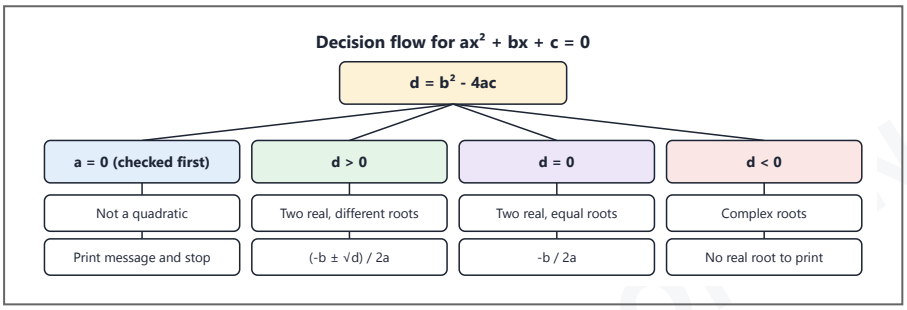
**Q.4 (OR) · Unit 2** 10 marks · ≈ 600–700 words

**Write C programs (i) to find the roots of a quadratic equation using the if-else-if ladder, (ii) to find the largest of three numbers using nested if and the conditional operator, and (iii) to check whether a character is a vowel or a consonant using switch. Explain each.**

## (I) ROOTS OF A QUADRATIC EQUATION

### Logic

For $ax^2 + bx + c = 0$ the discriminant is $d = b^2 - 4ac$. If $d > 0$ the roots are real and different, if $d = 0$ they are real and equal, and if $d < 0$ they are complex. The roots are $(-b \pm \sqrt{d}) / 2a$. Since three cases and the special case $a = 0$ exist, an if-else-if ladder is suitable.




*Decision flow for the roots of a quadratic equation*

### Program

```
#include <stdio.h>
#include <math.h>

int main(void)
{
    float a, b, c, d, r1, r2;
    printf("Enter a, b and c: ");
    scanf("%f %f %f", &a, &b, &c);
    if (a == 0) {
        printf("Not a quadratic equation\n");
        return 0;
    }
    d = b * b - 4 * a * c;
    if (d > 0) {
        r1 = (-b + sqrt(d)) / (2 * a);
        r2 = (-b - sqrt(d)) / (2 * a);
        printf("Real and different roots: %.2f and %.2f\n", r1, r2);
    } else if (d == 0) {
        r1 = -b / (2 * a);
        printf("Real and equal roots: %.2f and %.2f\n", r1, r1);
    } else {
        printf("Roots are complex (discriminant = %.2f)\n", d);
    }
    return 0;
}
```

**Output (gcc, compiled with -lm; inputs 1 -5 6, 1 2 1 and 1 2 5):**

```
Enter a, b and c: Real and different roots: 3.00 and 2.00
Enter a, b and c: Real and equal roots: -1.00 and -1.00
Enter a, b and c: Roots are complex (discriminant = -16.00)
```

The library function sqrt() needs math.h, and on Linux the program must be linked with -lm. Note also that comparing a float with == can fail because of rounding errors, so real programs often test

© Online Student Seva · Exclusive property · Accessible only through the Pariksha Pass App · Copying, sharing or selling is strictly prohibited and will invite serious legal action.
© ऑनलाइन स्टूडेंट सेवा · विशेष संपत्ति · केवल परीक्षा पास ऐप पर उपलब्ध · कॉपी, शेयर या बिक्री पर कड़ी कानूनी कार्रवाई होगी।

fabs(d) < 1e-6 instead of d == 0.

## (II) LARGEST OF THREE NUMBERS

### Logic

Compare a with b. The larger of the two is then compared with c. Two ways are shown: nested if statements, and the conditional operator (condition ? value1 : value2), which is a short form of if-else.

### Program

```
#include <stdio.h>

int main(void)
{
    int a, b, c, big;
    printf("Enter three numbers: ");
    scanf("%d %d %d", &a, &b, &c);

    if (a >= b) {
        if (a >= c)
            big = a;
        else
            big = c;
    } else {
        if (b >= c)
            big = b;
        else
            big = c;
    }
    printf("Largest (nested if)    = %d\n", big);

    big = (a > b) ? ((a > c) ? a : c) : ((b > c) ? b : c);
    printf("Largest (ternary)      = %d\n", big);
    return 0;
}
```

**Output (gcc; input 17 42 29):**

```
Enter three numbers: Largest (nested if)    = 42
Largest (ternary)      = 42
```

## (III) VOWEL OR CONSONANT USING SWITCH

### Logic

The character is converted to lower case with tolower(), so that both A and a are handled. Several case labels without break share one statement, which is called fall-through. Any other alphabet goes to default and is a consonant.

© Online Student Seva · Exclusive property · Accessible only through the Pariksha Pass App · Copying, sharing or selling is strictly prohibited and will invite serious legal action.
© ऑनलाइन स्टूडेंट सेवा · विशेष संपत्ति · केवल परीक्षा पास ऐप पर उपलब्ध · कॉपी, शेयर या बिक्री पर कड़ी कानूनी कार्रवाई होगी।

**Program**

```
#include <stdio.h>
#include <ctype.h>

int main(void)
{
    char ch;
    printf("Enter an alphabet: ");
    scanf(" %c", &ch);
    if (!isalpha(ch)) {
        printf("%c is not an alphabet\n", ch);
        return 0;
    }
    switch (tolower(ch)) {
        case 'a':
        case 'e':
        case 'i':
        case 'o':
        case 'u':
            printf("%c is a vowel\n", ch);
            break;
        default:
            printf("%c is a consonant\n", ch);
    }
    return 0;
}
```

**Output (gcc; inputs E, k and 7):**

```
Enter an alphabet: E is a vowel
Enter an alphabet: k is a consonant
Enter an alphabet: 7 is not an alphabet
```

### CONCLUSION

The if-else-if ladder suits several exclusive ranges of conditions, nested if and the conditional operator suit comparisons of values, and switch suits a single value matched against fixed constants. Every program should be tested with at least one input for each branch.

**Q.5 · Unit 3** 10 marks · ≈ 600–700 words

**What is searching? Explain linear search and binary search with algorithms, programs and outputs. Compare the two methods.**

### INTRODUCTION

Searching is the process of finding whether a given value, called the key, is present in a collection of data such as an array, and if so, at which position. The two basic methods for an array are linear search and binary search. The choice depends on whether the array is sorted and on its size.

### 1. LINEAR (SEQUENTIAL) SEARCH

**Algorithm**

1. Start from the first element, a[0].
2. Compare the key with the current element.
3. If they are equal, report the position and stop.

© Online Student Seva · Exclusive property · Accessible only through the Pariksha Pass App · Copying, sharing or selling is strictly prohibited and will invite serious legal action.
© ऑनलाइन स्टूडेंट सेवा · विशेष संपत्ति · केवल परीक्षा पास ऐप पर उपलब्ध · कॉपी, शेयर या बिक्री पर कड़ी कानूनी कार्रवाई होगी।

4. Otherwise move to the next element.
5. If the end of the array is reached, report that the key is not found.

It works on any array, sorted or unsorted. In the worst case, it makes n comparisons for n elements.

### Program

```
#include <stdio.h>

int main(void)
{
    int a[7] = {23, 45, 12, 67, 89, 34, 56};
    int i, key, pos = -1;
    printf("Enter the number to search: ");
    scanf("%d", &key);
    for (i = 0; i < 7; i++) {
        if (a[i] == key) {
            pos = i;
            break;
        }
    }
    if (pos >= 0)
        printf("%d found at position %d after %d comparisons\n", key, pos + 1, pos + 1);
    else
        printf("%d not found after 7 comparisons\n", key);
    return 0;
}
```

**Output (gcc; inputs 67 and 50):**

```
Enter the number to search: 67 found at position 4 after 4 comparisons
Enter the number to search: 50 not found after 7 comparisons
```

## 2. BINARY SEARCH

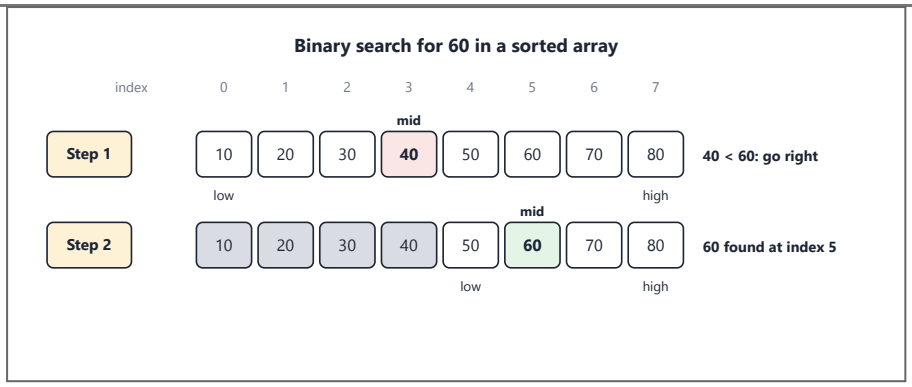
### Condition

The array must already be **sorted**. The method repeatedly halves the part of the array that can still contain the key.

### Algorithm

1. Set low = 0 and high = n - 1.
2. While low <= high, find mid = (low + high) / 2.
3. If a[mid] equals the key, the search is successful.
4. If the key is greater than a[mid], search the right half: low = mid + 1.
5. Otherwise search the left half: high = mid - 1.
6. If low becomes greater than high, the key is not present.

© Online Student Seva · Exclusive property · Accessible only through the Pariksha Pass App · Copying, sharing or selling is strictly prohibited and will invite serious legal action.
© ऑनलाइन स्टूडेंट सेवा · विशेष संपत्ति · केवल परीक्षा पास ऐप पर उपलब्ध · कॉपी, शेयर या बिक्री पर कड़ी कानूनी कार्रवाई होगी।




*Binary search for 60 in a sorted array*

### Program

```
#include <stdio.h>

int main(void)
{
    int a[8] = {10, 20, 30, 40, 50, 60, 70, 80};   /* sorted */
    int low = 0, high = 7, mid, key, steps = 0, found = 0;
    printf("Enter the number to search: ");
    scanf("%d", &key);
    while (low <= high) {
        mid = (low + high) / 2;
        steps++;
        printf("Step %d: low=%d high=%d mid=%d a[mid]=%d\n", steps, low, high, mid, a[mid]);
        if (a[mid] == key) {
            found = 1;
            break;
        } else if (key > a[mid])
            low = mid + 1;
        else
            high = mid - 1;
    }
    if (found)
        printf("%d found at index %d\n", key, mid);
    else
        printf("%d not found\n", key);
    return 0;
}
```

### Output (gcc; inputs 60 and 35):

```
Enter the number to search: Step 1: low=0 high=7 mid=3 a[mid]=40
Step 2: low=4 high=7 mid=5 a[mid]=60
60 found at index 5
Enter the number to search: Step 1: low=0 high=7 mid=3 a[mid]=40
Step 2: low=0 high=2 mid=1 a[mid]=20
Step 3: low=2 high=2 mid=2 a[mid]=30
35 not found
```

© Online Student Seva · Exclusive property · Accessible only through the Pariksha Pass App · Copying, sharing or selling is strictly prohibited and will invite serious legal action.
© ऑनलाइन स्टूडेंट सेवा · विशेष संपत्ति · केवल परीक्षा पास ऐप पर उपलब्ध · कॉपी, शेयर या बिक्री पर कड़ी कानूनी कार्रवाई होगी।

### COMPARISON OF THE TWO METHODS

| Point | Linear search | Binary search |
|---|---|---|
| Array must be sorted | No | Yes |
| Method | Checks elements one by one | Halves the search range |
| Worst-case comparisons | n | about $\log_2 n$ |
| For 1,000,000 elements | up to 1,000,000 | about 20 |
| Programming | Very simple | Slightly complex |
| Best for | Small or unsorted data | Large sorted data |

### CONCLUSION

Linear search is simple and works on any data, but it is slow for large arrays. Binary search needs sorted data, but it needs only about $\log_2 n$ comparisons, so it is much faster. Binary search also needs direct access to the middle element, so it suits arrays but not linked lists.

**Q.5 (OR) · Unit 3** 10 marks · ≈ 600–700 words

**How are strings stored in C? Write user-defined functions equivalent to strlen(), strcpy(), strcat() and strcmp() without using the string library, and demonstrate them with a program.**

### INTRODUCTION

In C a string is not a built-in type. It is a sequence of characters kept in consecutive bytes of a char array and followed by the **null character '\0'**, whose value is 0. For example, char s[] = "Data"; occupies 5 bytes: 'D', 'a', 't', 'a', '\0'. The null character marks the end, so every string function works by scanning characters until it meets '\0'. The standard library (string.h) provides strlen(), strcpy(), strcat() and strcmp(), but writing them ourselves shows how strings really work, and it is a common practical question.

### LOGIC OF THE FOUR FUNCTIONS

| Function | Library equivalent | Idea |
|---|---|---|
| my_strlen(s) | strlen | Count characters until '\0' |
| my_strcpy(d, s) | strcpy | Copy each character, including '\0' |
| my_strcat(d, s) | strcat | Go to the end of d, then copy s there |
| my_strcmp(a, b) | strcmp | Move while characters are equal; return the difference at the first mismatch |

© Online Student Seva · Exclusive property · Accessible only through the Pariksha Pass App · Copying, sharing or selling is strictly prohibited and will invite serious legal action.
© ऑनलाइन स्टूडेंट सेवा · विशेष संपत्ति · केवल परीक्षा पास ऐप पर उपलब्ध · कॉपी, शेयर या बिक्री पर कड़ी कानूनी कार्रवाई होगी।

### THE FUNCTIONS

```
int my_strlen(const char s[])
{
    int i = 0;
    while (s[i] != '\0')
        i++;
    return i;
}

void my_strcpy(char dest[], const char src[])
{
    int i = 0;
    while ((dest[i] = src[i]) != '\0')
        i++;
}

void my_strcat(char dest[], const char src[])
{
    int i = my_strlen(dest), j = 0;
    while ((dest[i] = src[j]) != '\0') {
        i++;
        j++;
    }
}

int my_strcmp(const char a[], const char b[])
{
    int i = 0;
    while (a[i] != '\0' && a[i] == b[i])
        i++;
    return a[i] - b[i];
}
```

In my_strcpy() the assignment is done inside the condition: the copied character is tested, so the loop stops after '\0' has also been copied.

### TEST PROGRAM

```
#include <stdio.h>

/* the four functions given above are placed here */

int main(void)
{
    char s1[30] = "Data", s2[30] = "Base", s3[30];
    printf("Length of s1   : %d\n", my_strlen(s1));
    my_strcpy(s3, s1);
    printf("Copy of s1     : %s\n", s3);
    my_strcat(s1, s2);
    printf("s1 + s2        : %s\n", s1);
    printf("strcmp(apple, apricot) = %d\n", my_strcmp("apple", "apricot"));
    printf("strcmp(ball, ball)     = %d\n", my_strcmp("ball", "ball"));
    return 0;
}
```

**Output (gcc):**

```
Length of s1   : 4
Copy of s1     : Data
s1 + s2        : DataBase
strcmp(apple, apricot) = -2
strcmp(ball, ball)     = 0
```

© Online Student Seva · Exclusive property · Accessible only through the Pariksha Pass App · Copying, sharing or selling is strictly prohibited and will invite serious legal action.
© ऑनलाइन स्टूडेंट सेवा · विशेष संपत्ति · केवल परीक्षा पास ऐप पर उपलब्ध · कॉपी, शेयर या बिक्री पर कड़ी कानूनी कार्रवाई होगी।

### POINTER VERSION OF MY_STRLEN()

An array name passed to a function becomes a pointer to its first element, so const char s[] and const char *s mean the same thing. The length can therefore be found by pointer subtraction.

```
int ptr_strlen(const char *s)
{
    const char *p = s;
    while (*p != '\0')
        p++;
    return (int)(p - s);
}
```

The pointer p moves to the null character, and p - s is the number of characters between the start and the end. For "Data" it returns 4.

### DRY RUN

- **my_strlen("Data"):** s[0] to s[3] are not '\0', so i becomes 4, and s[4] is '\0', so 4 is returned.
- **my_strcmp("apple", "apricot"):** 'a' = 'a' and 'p' = 'p', but the characters at index 2 are 'p' (112) and 'r' (114). The difference 112 - 114 = -2 is negative, so "apple" comes before "apricot".

### POINTS TO REMEMBER

1. Make sure the destination array of strcpy() and strcat() has room for the whole result, '\0' included; otherwise the program overwrites neighbouring memory.
2. The standard strcmp() guarantees only the sign of the result (negative, zero or positive), not the exact value, so the answer -2 depends on our own function.
3. Strings cannot be copied or compared with = and ==, because these work on the address of the array and not on its contents.

### CONCLUSION

Every C string is therefore a char array whose last used element is '\0'. All string operations are loops over characters that stop at '\0', and the library functions do exactly this work efficiently. For examination answers, the logic, the code and a short dry run together earn full marks.

**Q.6 · Unit 4** 10 marks · ≈ 600–700 words

**What is an array of structures? Explain how a pointer to a structure is used to pass a structure to a function. Write a program that stores the marks of three students, calculates the total and average and finds the topper.**

### INTRODUCTION

A structure groups variables of different types under one name, for example the roll number, name and marks of a student. When the same record is needed for many students, we create an **array of structures**, in which every element is a complete structure. It is the C way of building a small table or database in memory.

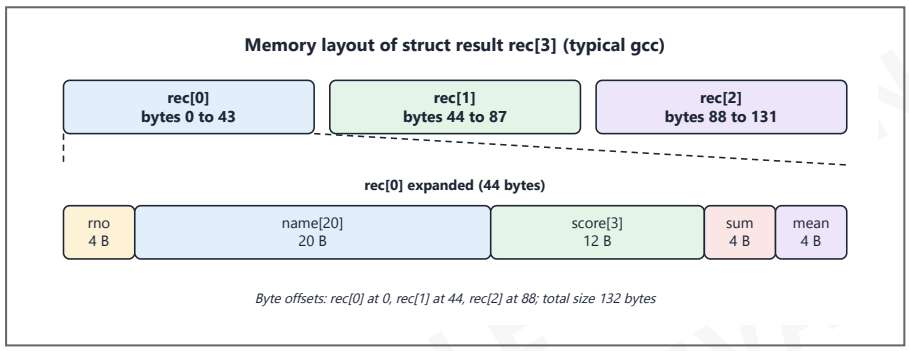
DEMO SAMPLE · ONLINE STUDENT SEVA

© Online Student Seva · Exclusive property · Accessible only through the Pariksha Pass App · Copying, sharing or selling is strictly prohibited and will invite serious legal action.
© ऑनलाइन स्टूडेंट सेवा · विशेष संपत्ति · केवल परीक्षा पास ऐप पर उपलब्ध · कॉपी, शेयर या बिक्री पर कड़ी कानूनी कार्रवाई होगी।

## DECLARATION AND ACCESS

```
struct result {
    int rno;
    char name[20];
    float score[3];
    float sum;
    float mean;
};
struct result rec[3];
```

- The statement struct result rec[3]; creates three structure variables, stored one after another in memory.
- A member is reached with the dot operator after the index: rec[1].rno, rec[1].score[2].
- On a typical gcc system each structure takes 44 bytes (4 + 20 + 12 + 4 + 4), so the array takes 132 bytes.




*Memory layout of an array of structures*

## POINTER TO A STRUCTURE

A pointer can hold the address of a structure: struct result *p = &rec[0];. Members are then accessed with the **arrow operator**: p->rno, which is the same as (*p).rno. Passing a pointer to a function gives it access to the original structure without copying all of its bytes, which is faster for large structures. Declaring the parameter as const protects the data from accidental change. Incrementing the pointer, p++, moves it to the next structure in the array.

© Online Student Seva · Exclusive property · Accessible only through the Pariksha Pass App · Copying, sharing or selling is strictly prohibited and will invite serious legal action.
© ऑनलाइन स्टूडेंट सेवा · विशेष संपत्ति · केवल परीक्षा पास ऐप पर उपलब्ध · कॉपी, शेयर या बिक्री पर कड़ी कानूनी कार्रवाई होगी।

## PROGRAM

```
#include <stdio.h>

struct result {
    int rno;
    char name[20];
    float score[3];
    float sum;
    float mean;
};

void show(const struct result *s)
{
    printf("%-4d %-8s %6.0f %7.2f\n", s->rno, s->name, s->sum, s->mean);
}

int main(void)
{
    struct result rec[3] = {
        {1, "Anil",   {78, 82, 69}},
        {2, "Bhavna", {91, 88, 94}},
        {3, "Chetan", {65, 70, 58}}
    };
    int i, j, top = 0;

    for (i = 0; i < 3; i++) {
        rec[i].sum = 0;
        for (j = 0; j < 3; j++)
            rec[i].sum += rec[i].score[j];
        rec[i].mean = rec[i].sum / 3;
        if (rec[i].sum > rec[top].sum)
            top = i;
    }
    printf("Roll Name     Total Average\n");
    for (i = 0; i < 3; i++)
        show(&rec[i]);
    printf("Topper: %s with %.0f marks\n", rec[top].name, rec[top].sum);
    return 0;
}
```

**Output (gcc):**

```
Roll Name     Total Average
1    Anil        229   76.33
2    Bhavna      273   91.00
3    Chetan      193   64.33
Topper: Bhavna with 273 marks
```

## EXPLANATION

1. The initialiser lists the roll number, the name and the three marks of each student.
2. The outer loop visits each student, and the inner loop adds the three marks into sum.
3. The average is sum / 3. The index top remembers the student with the highest total so far.
4. The function show() receives the address &rec[i] and prints the members through the arrow operator.
5. Totals: 78 + 82 + 69 = 229, 91 + 88 + 94 = 273 and 65 + 70 + 58 = 193, so Bhavna is the topper.

## COMMON MISTAKES

1. Forgetting the index: writing rec.rno instead of rec[0].rno.

© Online Student Seva · Exclusive property · Accessible only through the Pariksha Pass App · Copying, sharing or selling is strictly prohibited and will invite serious legal action.
© ऑनलाइन स्टूडेंट सेवा · विशेष संपत्ति · केवल परीक्षा पास ऐप पर उपलब्ध · कॉपी, शेयर या बिक्री पर कड़ी कानूनी कार्रवाई होगी।

2. Using the dot operator with a pointer, or the arrow operator with a variable; both are compile errors.
3. Comparing names with == instead of strcmp().
4. Using %d for float members in printf(), which prints a wrong value.

### WHY USE AN ARRAY OF STRUCTURES

The members of one record stay together, the records can be processed with a loop, the whole array can be passed to a function or sorted, and the same layout can be written to a file with fwrite().

### DOT OPERATOR AND ARROW OPERATOR

| Point | Dot (.) | Arrow (->) |
|---|---|---|
| Used with | Structure variable | Pointer to a structure |
| Example | rec[0].rno | p->rno |
| Equivalent | | (*p).rno |

### CONCLUSION

An array of structures stores many records of the same type, and a pointer to a structure lets a function use a record efficiently. Together they form the basis of simple record-processing programs.

**Q.6 (OR) · Unit 4** 10 marks · ≈ 600–700 words

**What is random access in files? Explain fseek(), ftell(), rewind() and the constants SEEK_SET, SEEK_CUR and SEEK_END. Write a program that stores records in a binary file and updates the second record in place.**

### INTRODUCTION

Files can be processed in two ways. In **sequential access** the data are read or written from the beginning to the end, one item after another. In **random access** the program moves directly to any byte of the file and reads or writes there, without reading the earlier data. Random access is needed when a single record has to be read or updated in a large file, for example one student or one stock item. C supports it by a **file position indicator**, which marks the position of the next read or write and can be moved by library functions.

### FUNCTIONS FOR RANDOM ACCESS

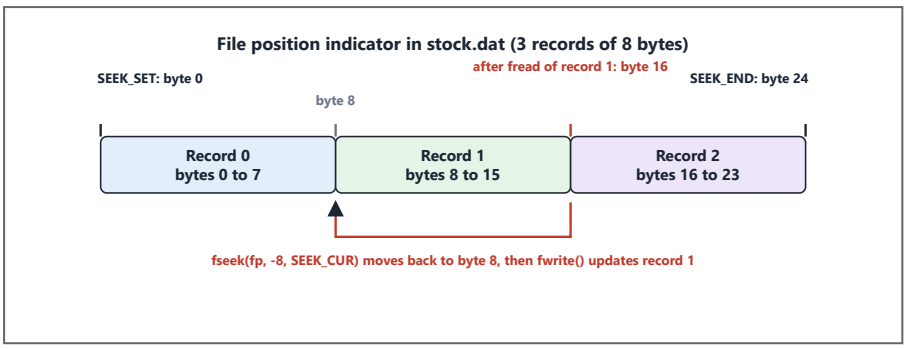
| Function | Purpose |
|---|---|
| fseek(fp, offset, origin) | Moves the position indicator by offset bytes from origin; returns 0 on success |
| ftell(fp) | Returns the current position, in bytes from the beginning, as a long |
| rewind(fp) | Moves the indicator to the beginning; same as fseek(fp, 0, SEEK_SET) |

The argument **origin** is one of three constants defined in stdio.h:

- **SEEK_SET**: the beginning of the file.
- **SEEK_CUR**: the current position.

© Online Student Seva · Exclusive property · Accessible only through the Pariksha Pass App · Copying, sharing or selling is strictly prohibited and will invite serious legal action.
© ऑनलाइन स्टूडेंट सेवा · विशेष संपत्ति · केवल परीक्षा पास ऐप पर उपलब्ध · कॉपी, शेयर या बिक्री पर कड़ी कानूनी कार्रवाई होगी।

- **SEEK_END**: the end of the file.




*File position indicator and the origins of fseek()*

Examples: fseek(fp, 0, SEEK_END) moves to the end, so ftell(fp) then gives the file size. fseek(fp, 2L*sizeof(struct item), SEEK_SET) moves to the start of the third record, since record numbering starts from 0. A negative offset, such as -1L*(long)sizeof(struct item) with SEEK_CUR, moves backwards by one record.

The mode "wb+" opens a binary file for both writing and reading, and creates it afresh. Random access is dependable in binary mode, because every record has a fixed size.

## PROGRAM

The program writes three records, finds the file size, reads the second record, adds 5 to its quantity, writes it back at the same place and prints all the records.

```
#include <stdio.h>

struct item {
    int code;
    int qty;
};

int main(void)
{
    struct item w[3] = {{101, 10}, {102, 25}, {103, 40}}, r;
    FILE *fp = fopen("stock.dat", "wb+");
    if (fp == NULL) {
        printf("Cannot open file\n");
        return 1;
    }
    fwrite(w, sizeof(struct item), 3, fp);

    fseek(fp, 0, SEEK_END);
    printf("File size = %ld bytes, records = %ld\n",
           ftell(fp), ftell(fp) / (long)sizeof(struct item));

    /* update the second record in place */
    fseek(fp, 1L * sizeof(struct item), SEEK_SET);
    fread(&r, sizeof(struct item), 1, fp);
    r.qty = r.qty + 5;
    fseek(fp, -1L * (long)sizeof(struct item), SEEK_CUR);
    fwrite(&r, sizeof(struct item), 1, fp);

    rewind(fp);
    while (fread(&r, sizeof(struct item), 1, fp) == 1)
        printf("Code %d  Qty %d\n", r.code, r.qty);
    fclose(fp);
    return 0;
}
```

© Online Student Seva · Exclusive property · Accessible only through the Pariksha Pass App · Copying, sharing or selling is strictly prohibited and will invite serious legal action.
© ऑनलाइन स्टूडेंट सेवा · विशेष संपत्ति · केवल परीक्षा पास ऐप पर उपलब्ध · कॉपी, शेयर या बिक्री पर कड़ी कानूनी कार्रवाई होगी।

**Output (gcc, where int is 4 bytes):**

```
File size = 24 bytes, records = 3
Code 101  Qty 10
Code 102  Qty 30
Code 103  Qty 40
```

## EXPLANATION

1. Each record has 8 bytes (two ints), so the three records make 24 bytes. After fseek(fp, 0, SEEK_END), ftell() returns 24 and 24 / 8 gives 3 records.
2. To reach the second record the indicator is moved 8 bytes from the beginning. fread() reads it, and the indicator is now at byte 16.
3. After the read, the indicator is behind the record, so fseek(..., -8, SEEK_CUR) is used to go back to byte 8 before writing. Without this step the new data would overwrite the third record.
4. rewind() returns to the start, and the loop reads until fread() fails at the end of the file.

## SEQUENTIAL AND RANDOM ACCESS

| Point | Sequential | Random |
|---|---|---|
| Movement | From start, in order | Directly to any byte |
| Functions | fgetc, fgets, fscanf, fread in a loop | fseek, ftell, rewind with fread, fwrite |
| Speed for one record | Slow in large files | Fast |

## CONCLUSION

fseek(), ftell() and rewind() control the file position indicator and make direct access to a record possible. In an update, the indicator must be moved back before writing, since every read or write advances it.

DEMO SAMPLE · ONLINE STUDENT SEVA

© Online Student Seva · Exclusive property · Accessible only through the Pariksha Pass App · Copying, sharing or selling is strictly prohibited and will invite serious legal action.
© ऑनलाइन स्टूडेंट सेवा · विशेष संपत्ति · केवल परीक्षा पास ऐप पर उपलब्ध · कॉपी, शेयर या बिक्री पर कड़ी कानूनी कार्रवाई होगी।

# भाग – II : हिन्दी माध्यम – आदर्श उत्तर

प्रत्येक उत्तर अंक-भार के अनुसार लिखा गया है — ढाँचा सीखें, फिर अपने शब्दों में लिखें।

## खण्ड – अ · प्रश्न 1 : उत्तर कुंजी

| क्र. | उत्तर | सही विकल्प | स्पष्टीकरण |
|---|---|---|---|
| 1 | (d) | 3 | दो पूर्णांकों के भाग का परिणाम पूर्णांक होता है और दशमलव भाग हट जाता है, इसलिए 7 / 2 = 3। 3.5 पाने के लिए एक संकार्य float होना चाहिए, जैसे 7 / 2.0। |
| 2 | (a) | \t | \t क्षैतिज टैब छापता है। \n नई पंक्ति, \b बैकस्पेस और \a ध्वनि संकेत (अलर्ट) है। |
| 3 | (d) | && | && तार्किक AND ऑपरेटर है। & बिटवाइज़ ऑपरेटर, == संबंधपरक (रिलेशनल) ऑपरेटर और ?: कंडीशनल ऑपरेटर है। |
| 4 | (a) | 0 | 5 > 3 सत्य (1) है पर 2 > 4 असत्य (0) है। व्यंजक 1 && 0 असत्य है और असत्य परिणाम 0 के रूप में छपता है। |
| 5 | (b) | वैकल्पिक (ऑप्शनल) है | default केस वैकल्पिक है। जब कोई भी case लेबल switch व्यंजक से मेल नहीं खाता तब यह चलता है। |
| 6 | (c) | दोनों स्ट्रिंग समान हों | strcmp() दोनों स्ट्रिंग समान होने पर 0, s1 < s2 होने पर ऋणात्मक मान और s1 > s2 होने पर धनात्मक मान लौटाता है। |
| 7 | (b) | आधार शर्त (बेस कंडीशन) | आधार शर्त फंक्शन को स्वयं को पुनः बुलाने से रोकती है। इसके बिना कॉल तब तक चलती रहती हैं जब तक स्टैक ओवरफ्लो न हो जाए। |
| 8 | (b) | स्ट्रक्चर के पॉइंटर से | डॉट ऑपरेटर स्ट्रक्चर वेरिएबल के साथ और तीर ऑपरेटर स्ट्रक्चर के पॉइंटर के साथ प्रयुक्त होता है: ptr->member और (*ptr).member समान हैं। |
| 9 | (c) | NULL | fopen() FILE पॉइंटर लौटाता है और विफल होने पर NULL लौटाता है। पढ़ने या लिखने से पहले प्रोग्राम को यह मान सदैव जाँचना चाहिए। |
| 10 | (c) | static | static स्थानीय वेरिएबल एक बार बनता है और प्रोग्राम समाप्त होने तक अपना मान रखता है। auto वेरिएबल हर कॉल पर नया बनता है। |

## खण्ड – अ · प्रश्न 2 : लघु उत्तर

**प्रश्न 2(a)** — 4 अंक · शब्द-सीमा ≈ 100–120 शब्द

### C में एस्केप सीक्वेंस क्या हैं? कोई पाँच अर्थ सहित लिखिए और एक उदाहरण दीजिए।

एस्केप सीक्वेंस बैकस्लैश से आरंभ होने वाले विशेष वर्ण-संयोजन हैं। ये ऐसे वर्णों, जैसे नई पंक्ति, को दर्शाते हैं जिन्हें स्ट्रिंग के भीतर सीधे टाइप नहीं किया जा सकता। कंपाइलर प्रत्येक सीक्वेंस को एक वर्ण मानता है।

| सीक्वेंस | अर्थ |
|---|---|
| \n | नई पंक्ति |
| \t | क्षैतिज टैब |
| \b | बैकस्पेस |
| \\ | बैकस्लैश |
| \" | डबल कोट |

**उदाहरण:**

© Online Student Seva · Exclusive property · Accessible only through the Pariksha Pass App · Copying, sharing or selling is strictly prohibited and will invite serious legal action.
© ऑनलाइन स्टूडेंट सेवा · विशेष संपत्ति · केवल परीक्षा पास ऐप पर उपलब्ध · कॉपी, शेयर या बिक्री पर कड़ी कानूनी कार्रवाई होगी।

```
printf("Name:\tAnil\n\"C\" is fun\n");
```

**आउटपुट (टैब को रिक्त स्थान से दिखाया गया):**

```
Name:    Anil
"C" is fun
```

यहाँ \t अगले टैब स्टॉप पर जाता है, \n नई पंक्ति आरंभ करता है, और \" स्ट्रिंग समाप्त किए बिना उद्धरण चिह्न छापता है। अन्य उपयोगी सीक्वेंस हैं: चेतावनी ध्वनि के लिए \a और स्ट्रिंग समाप्त करने वाले नल वर्ण के लिए \0।

**प्रश्न 2(b)** 4 अंक · शब्द-सीमा ≈ 100–120 शब्द

## for लूप का प्रयोग करके किसी संख्या की गुणन-सारणी (पहाड़ा) छापने का C प्रोग्राम लिखिए।

**तर्क:** संख्या को n में रखें और i = 1 से 10 तक गुणा n x i दोहराएँ, लूप के प्रत्येक चक्र में एक पंक्ति छापें।

```
#include <stdio.h>

int main(void)
{
    int n = 7, i;
    for (i = 1; i <= 10; i++)
        printf("%d x %2d = %2d\n", n, i, n * i);
    return 0;
}
```

**आउटपुट (पहली और अंतिम पंक्तियाँ):**

```
7 x  1 =  7
7 x  2 = 14
...
7 x 10 = 70
```

लूप वेरिएबल i, 1 से आरंभ होता है, प्रत्येक चक्र से पहले शर्त i <= 10 जाँची जाती है, और प्रत्येक चक्र के बाद i++ उसे बढ़ाता है, इसलिए लूप की बॉडी ठीक 10 बार चलती है। चौड़ाई %2d स्तंभों को संरेखित रखती है।

**प्रश्न 2(c)** 4 अंक · शब्द-सीमा ≈ 100–120 शब्द

## C में फंक्शन प्रोटोटाइप क्या है? यह क्यों आवश्यक है? उदाहरण सहित समझाइए।

फंक्शन प्रोटोटाइप (डिक्लेरेशन) फंक्शन के उपयोग से पहले कंपाइलर को फंक्शन का नाम, रिटर्न टाइप और पैरामीटर के टाइप बताता है। इसका रूप है: return_type name(type1, type2);

C प्रोग्राम को ऊपर से नीचे पढ़ता है। यदि main() अपने नीचे परिभाषित फंक्शन को बुलाता है, तो कंपाइलर को कॉल को जाँचने का तरीका पहले से पता होना चाहिए। प्रोटोटाइप न होने पर पुराने कंपाइलर int मान लेते हैं और नए कंपाइलर चेतावनी या त्रुटि देते हैं।

© Online Student Seva · Exclusive property · Accessible only through the Pariksha Pass App · Copying, sharing or selling is strictly prohibited and will invite serious legal action.
© ऑनलाइन स्टूडेंट सेवा · विशेष संपत्ति · केवल परीक्षा पास ऐप पर उपलब्ध · कॉपी, शेयर या बिक्री पर कड़ी कानूनी कार्रवाई होगी।

```
int add(int, int);                  /* प्रोटोटाइप */

int main(void)
{
    printf("%d", add(2, 3));        /* 5 छापता है */
    return 0;
}

int add(int a, int b)               /* डेफिनिशन */
{
    return a + b;
}
```

प्रोटोटाइप आर्ग्युमेंट की टाइप-जाँच संभव बनाता है और फंक्शन को main() के बाद परिभाषित करने देता है।

**प्रश्न 2(d)** — 4 अंक · शब्द-सीमा ≈ 100–120 शब्द

## C में void पॉइंटर और NULL पॉइंटर को उदाहरण सहित समझाइए।

**void पॉइंटर:** void पॉइंटर (void *p) सामान्य पॉइंटर है जो किसी भी डेटा टाइप के वेरिएबल का पता रख सकता है। इसे सीधे डीरेफरेंस नहीं किया जा सकता, क्योंकि डेटा का प्रकार और आकार अज्ञात होता है; पहले टाइप-कास्ट करना पड़ता है।

```
int x = 10;
void *p = &x;
printf("%d", *(int *)p);    /* 10 छापता है */
```

**NULL पॉइंटर:** NULL पॉइंटर किसी चीज़ की ओर संकेत नहीं करता। इसे p = NULL; लिखते हैं जहाँ NULL, stdio.h तथा stddef.h में मान 0 वाला मैक्रो है। इसे डीरेफरेंस करना त्रुटि है, इसलिए उपयोग से पहले if (p != NULL) से जाँचा जाता है। इसका उपयोग पॉइंटर के प्रारंभीकरण और लिंक्ड लिस्ट का अंत दर्शाने में होता है।

**प्रश्न 2(e)** — 4 अंक · शब्द-सीमा ≈ 100–120 शब्द

## C में टेक्स्ट फाइल तथा बाइनरी फाइल में अंतर कीजिए।

टेक्स्ट फाइल डेटा को पढ़ने योग्य वर्णों के रूप में रखती है, जबकि बाइनरी फाइल बाइटों को ठीक वैसे ही रखती है जैसे वे मेमोरी में होते हैं।

| बिंदु | टेक्स्ट फाइल | बाइनरी फाइल |
|---|---|---|
| सामग्री | पढ़ने योग्य वर्ण | कच्ची बाइटें, पढ़ने योग्य नहीं |
| मोड | "r", "w", "a" | "rb", "wb", "ab" |
| फंक्शन | fprintf, fscanf, fgets, fputs | fread, fwrite |
| न्यूलाइन | रूपांतरित, जैसे Windows पर \n से \r\n | रूपांतरित नहीं |
| आकार | संख्याओं के लिए बड़ा | संक्षिप्त |
| गति | धीमी, रूपांतरण आवश्यक | तेज, रूपांतरण नहीं |

उदाहरण के लिए, पूर्णांक 12345 टेक्स्ट फाइल में 5 बाइट (प्रति अंक एक) लेता है, जबकि सामान्य सिस्टम पर बाइनरी int के रूप में केवल 4 बाइट। स्ट्रक्चर के रिकॉर्डों के लिए बाइनरी फाइल और रिपोर्ट व कॉन्फ़िगरेशन

© Online Student Seva · Exclusive property · Accessible only through the Pariksha Pass App · Copying, sharing or selling is strictly prohibited and will invite serious legal action.
© ऑनलाइन स्टूडेंट सेवा · विशेष संपत्ति · केवल परीक्षा पास ऐप पर उपलब्ध · कॉपी, शेयर या बिक्री पर कड़ी कानूनी कार्रवाई होगी।

के लिए टेक्स्ट फाइल को वरीयता दी जाती है।

## खण्ड – ब · दीर्घ उत्तर

**प्रश्न 3 • इकाई 1** 10 अंक · ≈ 600–700 शब्द

**C में अनफॉर्मेटेड इनपुट और आउटपुट फंक्शन क्या हैं? getchar(), putchar(), gets(), puts() तथा fgets() को प्रोग्राम सहित समझाइए और इन्हें scanf() व printf() से अलग कीजिए।**

### भूमिका

C में इनपुट और आउटपुट हेडर फाइल stdio.h में घोषित लाइब्रेरी फंक्शनों से होता है। ये दो प्रकार के हैं। **फॉर्मेटेड** फंक्शन, printf() और scanf(), %d तथा %f जैसे फॉर्मेट स्पेसिफायर के अनुसार डेटा का रूपांतरण करते हैं। **अनफॉर्मेटेड** फंक्शन बिना रूपांतरण के कच्चे वर्ण या स्ट्रिंग पढ़ते या लिखते हैं। ये सरल और तेज़ हैं तथा पाठ को एक-एक वर्ण या एक-एक पंक्ति में संसाधित करने के लिए उपयोगी हैं।

### वर्ण इनपुट और आउटपुट

| फंक्शन | हेडर | उद्देश्य |
|---|---|---|
| getchar() | stdio.h | कीबोर्ड से एक वर्ण पढ़ता है (Enter की प्रतीक्षा करता है) |
| putchar(c) | stdio.h | स्क्रीन पर एक वर्ण लिखता है |
| getch() | conio.h | बिना इको और बिना Enter के एक वर्ण पढ़ता है |
| getche() | conio.h | इको के साथ, बिना Enter के एक वर्ण पढ़ता है |

getchar() वर्ण को int के रूप में लौटाता है ताकि विशेष मान EOF (फाइल का अंत) भी दर्शाया जा सके। getch() और getche() conio.h के हैं जो मानक C का भाग नहीं है: यह Turbo C और Windows कंपाइलरों में मिलता है, Linux के gcc में नहीं।

### स्ट्रिंग इनपुट और आउटपुट

- **puts(s)** स्ट्रिंग s लिखता है और स्वतः नई पंक्ति जोड़ देता है।
- **gets(s)** एक पंक्ति को s में पढ़ता है, पर ऐरे के आकार की जाँच कभी नहीं करता। इससे बफर ओवरफ्लो हो सकता है, इसलिए gets() को **C11 मानक से हटा दिया गया** और इसका उपयोग नहीं करना चाहिए।
- **fgets(s, n, stdin)** सुरक्षित विकल्प है। यह अधिकतम n - 1 वर्ण पढ़ता है, नई पंक्ति पर रुकता है, नई पंक्ति को स्ट्रिंग में रखता है और नल वर्ण जोड़ता है।
- **scanf("%s", s)** पहले रिक्त स्थान पर रुक जाता है, इसलिए पूरा नाम नहीं पढ़ सकता।

© Online Student Seva · Exclusive property · Accessible only through the Pariksha Pass App · Copying, sharing or selling is strictly prohibited and will invite serious legal action.
© ऑनलाइन स्टूडेंट सेवा · विशेष संपत्ति · केवल परीक्षा पास ऐप पर उपलब्ध · कॉपी, शेयर या बिक्री पर कड़ी कानूनी कार्रवाई होगी।

### प्रोग्राम 1: getchar() और putchar() से वर्ण संसाधन

```
#include <stdio.h>
#include <ctype.h>

int main(void)
{
    int ch, letters = 0, digits = 0, others = 0;
    printf("Type a line: ");
    while ((ch = getchar()) != '\n' && ch != EOF) {
        if (isalpha(ch))
            letters++;
        else if (isdigit(ch))
            digits++;
        else
            others++;
        putchar(toupper(ch));
    }
    putchar('\n');
    printf("Letters=%d Digits=%d Others=%d\n", letters, digits, others);
    return 0;
}
```

**आउटपुट (gcc; इनपुट Raipur 492001!):**

```
Type a line: RAIPUR 492001!
Letters=6 Digits=6 Others=2
```

लूप नई पंक्ति तक एक-एक वर्ण पढ़ता है, उसे वर्गीकृत करता है और उसका अपर-केस रूप छापता है। फंक्शन isalpha() और isdigit() ctype.h के हैं और वर्ण के अक्षर या अंक होने पर अशून्य मान लौटाते हैं। फंक्शन toupper() छोटे अक्षर को बड़े अक्षर में बदलता है और अन्य वर्णों को अपरिवर्तित रखता है।

### प्रोग्राम 2: fgets() और puts() से पूरी पंक्ति पढ़ना

```
#include <stdio.h>
#include <string.h>

int main(void)
{
    char name[30];
    printf("Enter your full name: ");
    fgets(name, sizeof(name), stdin);
    name[strcspn(name, "\n")] = '\0';
    puts("Hello,");
    puts(name);
    printf("Length = %d\n", (int)strlen(name));
    return 0;
}
```

**आउटपुट (gcc; इनपुट Anil Kumar Verma):**

```
Enter your full name: Hello,
Anil Kumar Verma
Length = 16
```

strcspn(name, "\n") नई पंक्ति की स्थिति खोजता है जिसे नल वर्ण से बदल दिया जाता है, इसलिए लंबाई में वह नहीं गिनी जाती।

© Online Student Seva · Exclusive property · Accessible only through the Pariksha Pass App · Copying, sharing or selling is strictly prohibited and will invite serious legal action.
© ऑनलाइन स्टूडेंट सेवा · विशेष संपत्ति · केवल परीक्षा पास ऐप पर उपलब्ध · कॉपी, शेयर या बिक्री पर कड़ी कानूनी कार्रवाई होगी।

### अनफॉर्मेटेड और फॉर्मेटेड फंक्शनों में अंतर

| बिंदु | अनफॉर्मेटेड (getchar, puts, fgets) | फॉर्मेटेड (scanf, printf) |
|---|---|---|
| रूपांतरण | कोई नहीं, कच्चे वर्ण | फॉर्मेट स्पेसिफायर प्रयुक्त |
| डेटा | वर्ण और स्ट्रिंग | सभी डेटा टाइप |
| इनपुट में रिक्त स्थान | fgets पढ़ लेता है | scanf वहाँ रुक जाता है |
| गति | तेज़ | धीमी |
| उपयोग | पाठ संसाधन | संख्यात्मक व मिश्रित इनपुट |

### एक सामान्य समस्या

scanf("%d", &n) के बाद उपयोगकर्ता द्वारा टाइप की गई नई पंक्ति इनपुट बफर में रह जाती है। इसके बाद का getchar() या fgets() उसी नई पंक्ति को तुरंत पढ़ लेता है और ऐसा लगता है कि वह छूट गया। इसे अतिरिक्त getchar() कॉल से उसे हटाकर सुलझाया जाता है।

### निष्कर्ष

अनफॉर्मेटेड फंक्शन वर्णों और पंक्तियों पर सरल व तेज़ नियंत्रण देते हैं। वर्ण-लूप के लिए getchar() व putchar(), पंक्तियों के लिए puts() व fgets() उपयुक्त हैं, और असुरक्षित होने के कारण gets() के स्थान पर fgets() प्रयोग करना चाहिए।

**प्रश्न 3 (अथवा) · इकाई 1** 10 अंक · ≈ 600–700 शब्द

## C में बिटवाइज़ ऑपरेटर क्या हैं? सभी छह बिटवाइज़ ऑपरेटरों को सत्यता सारणी और प्रोग्राम सहित समझाइए तथा विषम या सम की जाँच, अदला-बदली और बिट को सेट या क्लियर करने जैसे उनके अनुप्रयोग बताइए।

### भूमिका

बिटवाइज़ ऑपरेटर पूर्णांक संकार्यों (char, short, int, long) के अलग-अलग बिटों पर सीधे कार्य करते हैं। इन्हें float या double पर नहीं लगाया जा सकता। बाइनरी निरूपण पर कार्य करने के कारण ये तेज़ होते हैं और डिवाइस ड्राइवर, एम्बेडेड प्रोग्राम, फ्लैग, संपीड़न तथा एन्क्रिप्शन में बहुत प्रयुक्त होते हैं।

### छह बिटवाइज़ ऑपरेटर

| ऑपरेटर | नाम | नियम |
|---|---|---|
| & | AND | परिणाम बिट 1 केवल तब जब दोनों बिट 1 हों |
| ऊर्ध्व रेखा | OR | परिणाम बिट 1 यदि कम से कम एक बिट 1 हो |
| ^ | XOR | परिणाम बिट 1 यदि बिट भिन्न हों |
| ~ | NOT (पूरक) | प्रत्येक बिट उलट देता है (एकल) |
| << | बायाँ शिफ्ट | बिट बाईं ओर खिसकाता है; प्रत्येक शिफ्ट से 2 से गुणा |
| >> | दायाँ शिफ्ट | बिट दाईं ओर खिसकाता है; प्रत्येक शिफ्ट से 2 से भाग |

### सत्यता सारणी

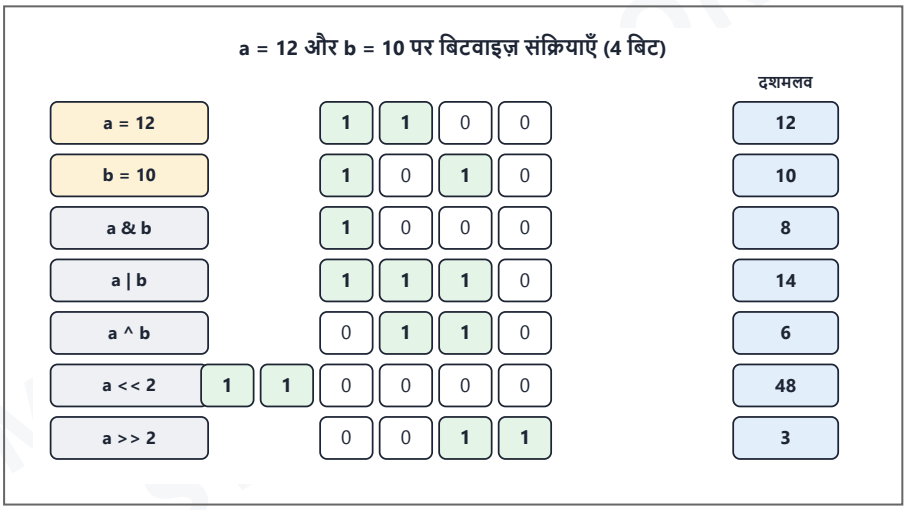
© Online Student Seva · Exclusive property · Accessible only through the Pariksha Pass App · Copying, sharing or selling is strictly prohibited and will invite serious legal action.
© ऑनलाइन स्टूडेंट सेवा · विशेष संपत्ति · केवल परीक्षा पास ऐप पर उपलब्ध · कॉपी, शेयर या बिक्री पर कड़ी कानूनी कार्रवाई होगी।

| a | b | a AND b | a OR b | a XOR b |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 | 0 |

## हल किया हुआ उदाहरण

मान लें 4 बिट में a = 12 (1100) और b = 10 (1010) है।

- a & b = 1000 = 8
- a | b = 1110 = 14
- a ^ b = 0110 = 6
- ~a सभी बिट उलट देता है। द्विपूरक (two's complement) वाली मशीन पर ~n = -(n + 1), इसलिए ~12 = -13।
- a << 2 = 110000 = 48, जो 12 x $2^2$ है।
- a >> 2 = 0011 = 3, जो 12 / $2^2$ है (धनात्मक संख्याओं के लिए)।




*a = 12 और b = 10 पर बिटवाइज़ संक्रियाएँ*

## प्रोग्राम 1: सभी ऑपरेटर

```
#include <stdio.h>

int main(void)
{
    int a = 12, b = 10;             /* 1100 and 1010 */
    printf("a & b  = %d\n", a & b);
    printf("a | b  = %d\n", a | b);
    printf("a ^ b  = %d\n", a ^ b);
    printf("~a     = %d\n", ~a);
    printf("a << 2 = %d\n", a << 2);
    printf("a >> 2 = %d\n", a >> 2);
    return 0;
}
```

**आउटपुट (gcc):**

© Online Student Seva · Exclusive property · Accessible only through the Pariksha Pass App · Copying, sharing or selling is strictly prohibited and will invite serious legal action.
© ऑनलाइन स्टूडेंट सेवा · विशेष संपत्ति · केवल परीक्षा पास ऐप पर उपलब्ध · कॉपी, शेयर या बिक्री पर कड़ी कानूनी कार्रवाई होगी।

```
a & b  = 8
a | b  = 14
a ^ b  = 6
~a     = -13
a << 2 = 48
a >> 2 = 3
```

### अनुप्रयोग

1. **विषम या सम:** विषम संख्या का अंतिम बिट 1 होता है, इसलिए विषम संख्याओं के लिए n & 1 अशून्य होता है।
2. **तीसरे वेरिएबल के बिना अदला-बदली:** a = a ^ b; b = a ^ b; a = a ^ b; क्योंकि x ^ x = 0 और x ^ 0 = x।
3. **2 की घात:** धनात्मक n 2 की घात है यदि n & (n - 1) शून्य हो, क्योंकि केवल एक बिट सेट होता है।
4. **बिट सेट करना:** x |= (1 << k)।
5. **बिट क्लियर करना:** x &= ~(1 << k)।
6. **बिट टॉगल करना:** x ^= (1 << k)।

### प्रोग्राम 2: अनुप्रयोग

```c
#include <stdio.h>

int main(void)
{
    int x = 25, p = 5, q = 9, n = 64;
    unsigned char r = 0;

    printf("%d is %s\n", x, (x & 1) ? "odd" : "even");
    p = p ^ q;
    q = p ^ q;
    p = p ^ q;
    printf("After swap: p = %d, q = %d\n", p, q);
    printf("%d %s a power of two\n", n,
           (n > 0 && (n & (n - 1)) == 0) ? "is" : "is not");
    r |= (1 << 3);                  /* set bit 3    */
    printf("After set    : %d\n", r);
    r ^= (1 << 0);                  /* toggle bit 0 */
    printf("After toggle : %d\n", r);
    r &= ~(1 << 3);                 /* clear bit 3  */
    printf("After clear  : %d\n", r);
    return 0;
}
```

**आउटपुट (gcc):**

```
25 is odd
After swap: p = 9, q = 5
64 is a power of two
After set    : 8
After toggle : 9
After clear  : 1
```

### याद रखने योग्य बातें

- & और | बिटवाइज़ हैं, जबकि && और || तार्किक ऑपरेटर हैं; इन्हें आपस में नहीं मिलाना चाहिए।
- ऋणात्मक साइन्ड संख्या का दायाँ शिफ्ट इम्प्लीमेंटेशन-डिफ़ाइंड है।

© Online Student Seva · Exclusive property · Accessible only through the Pariksha Pass App · Copying, sharing or selling is strictly prohibited and will invite serious legal action.
© ऑनलाइन स्टूडेंट सेवा · विशेष संपत्ति · केवल परीक्षा पास ऐप पर उपलब्ध · कॉपी, शेयर या बिक्री पर कड़ी कानूनी कार्रवाई होगी।

### निष्कर्ष

बिटवाइज़ ऑपरेटर डेटा को बिट स्तर पर संचालित करते हैं। 1 << k जैसे मास्क के साथ ये अलग-अलग बिटों को सेट, क्लियर और जाँचने देते हैं, तथा अदला-बदली और 2 की घात की जाँच जैसी समस्याओं के संक्षिप्त हल देते हैं।

**प्रश्न 4 · इकाई 2** 10 अंक · ≈ 600–700 शब्द

**C प्रोग्राम लिखिए (i) दशमलव संख्या को बाइनरी में बदलने का, (ii) यूक्लिड की विधि से दो संख्याओं का महत्तम समापवर्तक (GCD) और लघुत्तम समापवर्त्य (LCM) निकालने का, तथा (iii) 500 तक की पूर्ण संख्याएँ छापने का। प्रत्येक का तर्क समझाइए।**

### (i) दशमलव से बाइनरी

#### तर्क

संख्या को बार-बार 2 से भाग दें और प्रत्येक शेषफल को संग्रहीत करें। शेषफलों को उलटे क्रम (अंतिम पहले) में पढ़ने पर बाइनरी संख्या बनती है। लूप इसलिए प्रयुक्त होता है क्योंकि भागों की संख्या पहले से ज्ञात नहीं होती।

#### प्रोग्राम

```
#include <stdio.h>

int main(void)
{
    int n, bin[32], i = 0;
    printf("Enter a decimal number: ");
    scanf("%d", &n);
    printf("Binary of %d = ", n);
    if (n == 0)
        printf("0");
    while (n > 0) {
        bin[i] = n % 2;
        n = n / 2;
        i++;
    }
    for (i = i - 1; i >= 0; i--)
        printf("%d", bin[i]);
    printf("\n");
    return 0;
}
```

**आउटपुट (gcc; इनपुट 45 और 10):**

```
Enter a decimal number: Binary of 45 = 101101
Enter a decimal number: Binary of 10 = 1010
```

45 के लिए ड्राई रन: शेषफल 1, 0, 1, 1, 0, 1 इसी क्रम में मिलते हैं, और उन्हें उलटा छापने पर 101101 आता है।

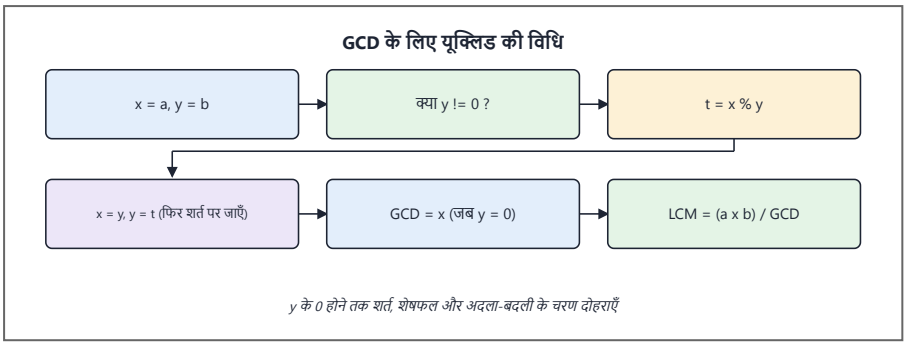
### (ii) यूक्लिड की विधि से GCD और LCM

#### तर्क

x और y का GCD वही है जो y और x % y का GCD है। यह चरण तब तक दोहराया जाता है जब तक शेषफल 0 न हो जाए, और अंतिम अशून्य संख्या GCD होती है। फिर LCM संबंध LCM = (a x b) / GCD से निकाला

© Online Student Seva · Exclusive property · Accessible only through the Pariksha Pass App · Copying, sharing or selling is strictly prohibited and will invite serious legal action.
© ऑनलाइन स्टूडेंट सेवा · विशेष संपत्ति · केवल परीक्षा पास ऐप पर उपलब्ध · कॉपी, शेयर या बिक्री पर कड़ी कानूनी कार्रवाई होगी।

जाता है।




*GCD के लिए यूक्लिड की विधि का फ्लोचार्ट*

### प्रोग्राम

```
#include <stdio.h>

int main(void)
{
    int a, b, x, y, t;
    printf("Enter two numbers: ");
    scanf("%d %d", &a, &b);
    x = a;
    y = b;
    while (y != 0) {
        t = x % y;
        x = y;
        y = t;
    }
    printf("GCD = %d\n", x);
    printf("LCM = %d\n", (a * b) / x);
    return 0;
}
```

### आउटपुट (gcc; इनपुट 48 36 और 17 5):

```
Enter two numbers: GCD = 12
LCM = 144
Enter two numbers: GCD = 1
LCM = 85
```

48 और 36 के लिए ड्राई रन: (x, y) = (48, 36), फिर (36, 12), फिर (12, 0)। लूप रुक जाता है और GCD 12 है, अतः LCM = 48 x 36 / 12 = 144।

## (iii) 500 तक की पूर्ण संख्याएँ

### तर्क

कोई संख्या पूर्ण होती है यदि वह अपने उचित भाजकों (स्वयं के अतिरिक्त भाजकों) के योग के बराबर हो। उदाहरण: 6 = 1 + 2 + 3 और 28 = 1 + 2 + 4 + 7 + 14। बाहरी लूप प्रत्येक संख्या n लेता है और भीतरी लूप 1 से n/2 तक सभी भाजक जोड़ता है, क्योंकि कोई उचित भाजक n/2 से बड़ा नहीं हो सकता।

© Online Student Seva · Exclusive property · Accessible only through the Pariksha Pass App · Copying, sharing or selling is strictly prohibited and will invite serious legal action.
© ऑनलाइन स्टूडेंट सेवा · विशेष संपत्ति · केवल परीक्षा पास ऐप पर उपलब्ध · कॉपी, शेयर या बिक्री पर कड़ी कानूनी कार्रवाई होगी।

### प्रोग्राम

```
#include <stdio.h>

int main(void)
{
    int n, i, sum;
    printf("Perfect numbers up to 500: ");
    for (n = 2; n <= 500; n++) {
        sum = 0;
        for (i = 1; i <= n / 2; i++) {
            if (n % i == 0)
                sum = sum + i;
        }
        if (sum == n)
            printf("%d ", n);
    }
    printf("\n");
    return 0;
}
```

**आउटपुट (gcc):**

```
Perfect numbers up to 500: 6 28 496
```

### टिप्पणियाँ

1. ऐरे bin[32] पर्याप्त बड़ी है क्योंकि int में अधिकतम 32 बिट होते हैं।
2. GCD प्रोग्राम में a और b को x और y में कॉपी किया गया है ताकि LCM सूत्र के लिए मूल मान उपलब्ध रहें।
3. बड़े इनपुट के लिए गुणनफल a x b ओवरफ्लो हो सकता है, इसलिए बड़ी संख्याओं के लिए LCM को (a / gcd) x b से निकालना बेहतर है।

### तीनों प्रोग्रामों का सारांश

| प्रोग्राम | मुख्य विचार | प्रयुक्त लूप |
|---|---|---|
| दशमलव से बाइनरी | 2 से बार-बार भाग | while, फिर for (उलटा) |
| GCD और LCM | gcd(x, y) = gcd(y, x % y) | while |
| पूर्ण संख्याएँ | उचित भाजकों का योग | नेस्टेड for |

### निष्कर्ष

तीनों प्रोग्राम लूप के भीतर % और / ऑपरेटरों पर निर्भर हैं। ड्राई रन और आउटपुट दिखाने से तर्क प्रमाणित होता है, और यूक्लिड की विधि हर भाजक की जाँच करने से कहीं तेज़ है।

**प्रश्न 4 (अथवा) • इकाई 2** 10 अंक · ≈ 600–700 शब्द

**C प्रोग्राम लिखिए (i) if-else-if सीढ़ी से द्विघात समीकरण के मूल निकालने का, (ii) नेस्टेड if और कंडीशनल ऑपरेटर से तीन संख्याओं में सबसे बड़ी संख्या निकालने का, तथा (iii) switch से जाँचने का**

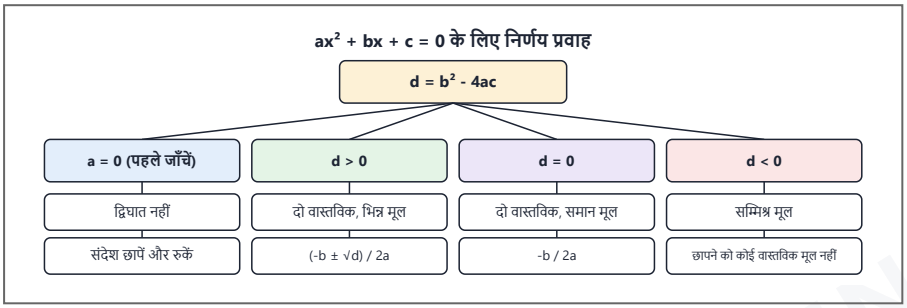
© Online Student Seva · Exclusive property · Accessible only through the Pariksha Pass App · Copying, sharing or selling is strictly prohibited and will invite serious legal action.
© ऑनलाइन स्टूडेंट सेवा · विशेष संपत्ति · केवल परीक्षा पास ऐप पर उपलब्ध · कॉपी, शेयर या बिक्री पर कड़ी कानूनी कार्रवाई होगी।

**कि कोई वर्ण स्वर है या व्यंजन। प्रत्येक को समझाइए।**

### (i) द्विघात समीकरण के मूल

#### तर्क

$ax^2 + bx + c = 0$ के लिए विविक्तकर $d = b^2 - 4ac$ होता है। यदि $d > 0$ तो मूल वास्तविक और भिन्न, $d = 0$ तो वास्तविक और समान, तथा $d < 0$ तो मूल सम्मिश्र होते हैं। मूल $(-b \pm \sqrt{d}) / 2a$ हैं। तीन स्थितियाँ और विशेष स्थिति $a = 0$ होने के कारण if-else-if सीढ़ी उपयुक्त है।




*द्विघात समीकरण के मूलों के लिए निर्णय प्रवाह*

#### प्रोग्राम

```
#include <stdio.h>
#include <math.h>

int main(void)
{
    float a, b, c, d, r1, r2;
    printf("Enter a, b and c: ");
    scanf("%f %f %f", &a, &b, &c);
    if (a == 0) {
        printf("Not a quadratic equation\n");
        return 0;
    }
    d = b * b - 4 * a * c;
    if (d > 0) {
        r1 = (-b + sqrt(d)) / (2 * a);
        r2 = (-b - sqrt(d)) / (2 * a);
        printf("Real and different roots: %.2f and %.2f\n", r1, r2);
    } else if (d == 0) {
        r1 = -b / (2 * a);
        printf("Real and equal roots: %.2f and %.2f\n", r1, r1);
    } else {
        printf("Roots are complex (discriminant = %.2f)\n", d);
    }
    return 0;
}
```

#### आउटपुट (gcc, -lm के साथ कंपाइल; इनपुट 1 -5 6, 1 2 1 और 1 2 5):

```
Enter a, b and c: Real and different roots: 3.00 and 2.00
Enter a, b and c: Real and equal roots: -1.00 and -1.00
Enter a, b and c: Roots are complex (discriminant = -16.00)
```

लाइब्रेरी फंक्शन sqrt() के लिए math.h चाहिए, और Linux पर प्रोग्राम को -lm के साथ लिंक करना पड़ता है। यह भी ध्यान रहे कि float की == से तुलना पूर्णन (राउंडिंग) त्रुटि के कारण विफल हो सकती है, इसलिए वास्तविक प्रोग्राम प्रायः $d == 0$ के स्थान पर $fabs(d) < 1e\text{-}6$ जाँचते हैं।

© Online Student Seva · Exclusive property · Accessible only through the Pariksha Pass App · Copying, sharing or selling is strictly prohibited and will invite serious legal action.
© ऑनलाइन स्टूडेंट सेवा · विशेष संपत्ति · केवल परीक्षा पास ऐप पर उपलब्ध · कॉपी, शेयर या बिक्री पर कड़ी कानूनी कार्रवाई होगी।

### (ii) तीन संख्याओं में सबसे बड़ी

**तर्क**

a की b से तुलना करें। फिर इन दोनों में बड़ी संख्या की c से तुलना करें। दो तरीके दिखाए गए हैं: नेस्टेड if स्टेटमेंट, और कंडीशनल ऑपरेटर (शर्त ? मान1 : मान2), जो if-else का संक्षिप्त रूप है।

**प्रोग्राम**

```
#include <stdio.h>

int main(void)
{
    int a, b, c, big;
    printf("Enter three numbers: ");
    scanf("%d %d %d", &a, &b, &c);

    if (a >= b) {
        if (a >= c)
            big = a;
        else
            big = c;
    } else {
        if (b >= c)
            big = b;
        else
            big = c;
    }
    printf("Largest (nested if)    = %d\n", big);

    big = (a > b) ? ((a > c) ? a : c) : ((b > c) ? b : c);
    printf("Largest (ternary)      = %d\n", big);
    return 0;
}
```

**आउटपुट (gcc; इनपुट 17 42 29):**

```
Enter three numbers: Largest (nested if)    = 42
Largest (ternary)      = 42
```

### (iii) switch से स्वर या व्यंजन

**तर्क**

वर्ण को tolower() से छोटे अक्षर में बदला जाता है ताकि A और a दोनों संभल जाएँ। बिना break वाले अनेक case लेबल एक ही स्टेटमेंट साझा करते हैं, जिसे फॉल-थ्रू कहते हैं। कोई अन्य अक्षर default में जाता है और व्यंजन है।

© Online Student Seva · Exclusive property · Accessible only through the Pariksha Pass App · Copying, sharing or selling is strictly prohibited and will invite serious legal action.
© ऑनलाइन स्टूडेंट सेवा · विशेष संपत्ति · केवल परीक्षा पास ऐप पर उपलब्ध · कॉपी, शेयर या बिक्री पर कड़ी कानूनी कार्रवाई होगी।

### प्रोग्राम

```
#include <stdio.h>
#include <ctype.h>

int main(void)
{
    char ch;
    printf("Enter an alphabet: ");
    scanf(" %c", &ch);
    if (!isalpha(ch)) {
        printf("%c is not an alphabet\n", ch);
        return 0;
    }
    switch (tolower(ch)) {
        case 'a':
        case 'e':
        case 'i':
        case 'o':
        case 'u':
            printf("%c is a vowel\n", ch);
            break;
        default:
            printf("%c is a consonant\n", ch);
    }
    return 0;
}
```

### आउटपुट (gcc; इनपुट E, k और 7):

```
Enter an alphabet: E is a vowel
Enter an alphabet: k is a consonant
Enter an alphabet: 7 is not an alphabet
```

### निष्कर्ष

if-else-if सीढ़ी कई परस्पर अपवर्जी शर्तों के लिए, नेस्टेड if और कंडीशनल ऑपरेटर मानों की तुलना के लिए, तथा switch एक मान को निश्चित स्थिरांकों से मिलाने के लिए उपयुक्त है। प्रत्येक प्रोग्राम को हर शाखा के लिए कम से कम एक इनपुट से परखना चाहिए।

**प्रश्न 5 • इकाई 3** 10 अंक · ≈ 600–700 शब्द

## खोज (सर्चिंग) क्या है? रैखिक खोज (लीनियर सर्च) और द्विआधारी खोज (बाइनरी सर्च) को एल्गोरिदम, प्रोग्राम और आउटपुट सहित समझाइए। दोनों विधियों की तुलना कीजिए।

### भूमिका

खोज (सर्चिंग) यह पता लगाने की प्रक्रिया है कि दिया गया मान, जिसे कुंजी (की) कहते हैं, ऐरे जैसे डेटा संग्रह में उपस्थित है या नहीं, और यदि है तो किस स्थान पर। ऐरे के लिए दो मूल विधियाँ रैखिक खोज और द्विआधारी खोज हैं। चुनाव इस पर निर्भर करता है कि ऐरे क्रमबद्ध है या नहीं और उसका आकार कितना है।

### 1. रैखिक (अनुक्रमिक) खोज

#### एल्गोरिदम

1. पहले तत्व a[0] से आरंभ करें।
2. कुंजी की वर्तमान तत्व से तुलना करें।

© Online Student Seva · Exclusive property · Accessible only through the Pariksha Pass App · Copying, sharing or selling is strictly prohibited and will invite serious legal action.
© ऑनलाइन स्टूडेंट सेवा · विशेष संपत्ति · केवल परीक्षा पास ऐप पर उपलब्ध · कॉपी, शेयर या बिक्री पर कड़ी कानूनी कार्रवाई होगी।

3. यदि बराबर हों तो स्थान बताएँ और रुक जाएँ।
4. अन्यथा अगले तत्व पर जाएँ।
5. यदि ऐरे का अंत आ जाए तो बताएँ कि कुंजी नहीं मिली।

यह किसी भी ऐरे पर, क्रमबद्ध या अक्रमबद्ध, कार्य करती है। n तत्वों के लिए सबसे बुरी स्थिति में यह n तुलनाएँ करती है।

### प्रोग्राम

```
#include <stdio.h>

int main(void)
{
    int a[7] = {23, 45, 12, 67, 89, 34, 56};
    int i, key, pos = -1;
    printf("Enter the number to search: ");
    scanf("%d", &key);
    for (i = 0; i < 7; i++) {
        if (a[i] == key) {
            pos = i;
            break;
        }
    }
    if (pos >= 0)
        printf("%d found at position %d after %d comparisons\n", key, pos + 1, pos + 1);
    else
        printf("%d not found after 7 comparisons\n", key);
    return 0;
}
```

### आउटपुट (gcc; इनपुट 67 और 50):

```
Enter the number to search: 67 found at position 4 after 4 comparisons
Enter the number to search: 50 not found after 7 comparisons
```

## 2. द्विआधारी खोज

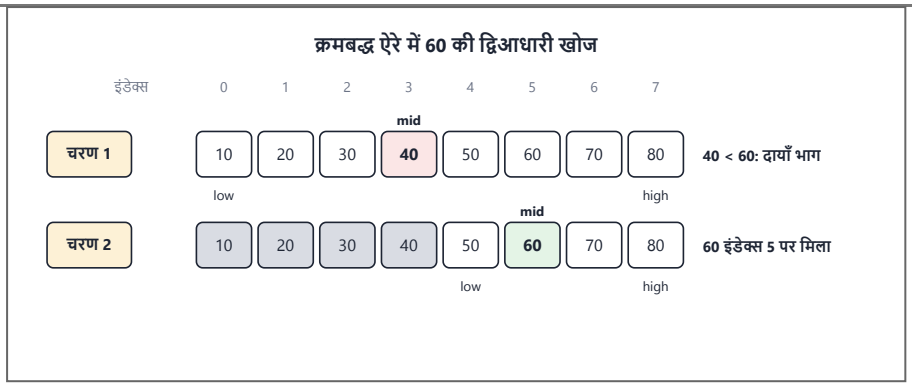
### शर्त

ऐरे पहले से **क्रमबद्ध** होना चाहिए। यह विधि ऐरे के उस भाग को बार-बार आधा करती है जिसमें कुंजी अब भी हो सकती है।

### एल्गोरिदम

1. low = 0 और high = n - 1 रखें।
2. जब तक low <= high है, mid = (low + high) / 2 निकालें।
3. यदि a[mid] कुंजी के बराबर है तो खोज सफल है।
4. यदि कुंजी a[mid] से बड़ी है तो दायाँ भाग खोजें: low = mid + 1।
5. अन्यथा बायाँ भाग खोजें: high = mid - 1।
6. यदि low, high से बड़ा हो जाए तो कुंजी उपस्थित नहीं है।

© Online Student Seva · Exclusive property · Accessible only through the Pariksha Pass App · Copying, sharing or selling is strictly prohibited and will invite serious legal action.
© ऑनलाइन स्टूडेंट सेवा · विशेष संपत्ति · केवल परीक्षा पास ऐप पर उपलब्ध · कॉपी, शेयर या बिक्री पर कड़ी कानूनी कार्रवाई होगी।




*क्रमबद्ध ऐरे में 60 की द्विआधारी खोज*

## प्रोग्राम

```
#include <stdio.h>

int main(void)
{
    int a[8] = {10, 20, 30, 40, 50, 60, 70, 80};   /* sorted */
    int low = 0, high = 7, mid, key, steps = 0, found = 0;
    printf("Enter the number to search: ");
    scanf("%d", &key);
    while (low <= high) {
        mid = (low + high) / 2;
        steps++;
        printf("Step %d: low=%d high=%d mid=%d a[mid]=%d\n", steps, low, high, mid, a[mid]);
        if (a[mid] == key) {
            found = 1;
            break;
        } else if (key > a[mid])
            low = mid + 1;
        else
            high = mid - 1;
    }
    if (found)
        printf("%d found at index %d\n", key, mid);
    else
        printf("%d not found\n", key);
    return 0;
}
```

## आउटपुट (gcc; इनपुट 60 और 35):

```
Enter the number to search: Step 1: low=0 high=7 mid=3 a[mid]=40
Step 2: low=4 high=7 mid=5 a[mid]=60
60 found at index 5
Enter the number to search: Step 1: low=0 high=7 mid=3 a[mid]=40
Step 2: low=0 high=2 mid=1 a[mid]=20
Step 3: low=2 high=2 mid=2 a[mid]=30
35 not found
```

© Online Student Seva · Exclusive property · Accessible only through the Pariksha Pass App · Copying, sharing or selling is strictly prohibited and will invite serious legal action.
© ऑनलाइन स्टूडेंट सेवा · विशेष संपत्ति · केवल परीक्षा पास ऐप पर उपलब्ध · कॉपी, शेयर या बिक्री पर कड़ी कानूनी कार्रवाई होगी।

### दोनों विधियों की तुलना

| बिंदु | रैखिक खोज | द्विआधारी खोज |
|---|---|---|
| ऐरे का क्रमबद्ध होना | आवश्यक नहीं | आवश्यक |
| विधि | तत्वों को एक-एक करके जाँचती है | खोज-परास को आधा करती है |
| सबसे बुरी स्थिति में तुलनाएँ | n | लगभग $\log_2 n$ |
| 10,00,000 तत्वों के लिए | 10,00,000 तक | लगभग 20 |
| प्रोग्रामिंग | बहुत सरल | थोड़ी जटिल |
| सर्वश्रेष्ठ | छोटा या अक्रमबद्ध डेटा | बड़ा क्रमबद्ध डेटा |

### निष्कर्ष

रैखिक खोज सरल है और किसी भी डेटा पर चलती है, किंतु बड़े ऐरे के लिए धीमी है। द्विआधारी खोज को क्रमबद्ध डेटा चाहिए, किंतु इसमें केवल लगभग $\log_2 n$ तुलनाएँ लगती हैं, इसलिए यह बहुत तेज़ है। द्विआधारी खोज को मध्य तत्व तक सीधी पहुँच भी चाहिए, इसलिए यह ऐरे के लिए उपयुक्त है, लिंक्ड लिस्ट के लिए नहीं।

**प्रश्न 5 (अथवा) · इकाई 3** 10 अंक · ≈ 600–700 शब्द

## C में स्ट्रिंग कैसे संग्रहीत होती हैं? स्ट्रिंग लाइब्रेरी का उपयोग किए बिना strlen(), strcpy(), strcat() तथा strcmp() के समतुल्य यूज़र-डिफ़ाइंड फंक्शन लिखिए और प्रोग्राम से उन्हें प्रदर्शित कीजिए।

### भूमिका

C में स्ट्रिंग कोई अंतर्निहित (बिल्ट-इन) टाइप नहीं है। यह char ऐरे के क्रमागत बाइटों में रखे वर्णों का अनुक्रम है जिसके बाद **नल वर्ण '\0'** आता है, जिसका मान 0 है। उदाहरण के लिए char s[] = "Data"; 5 बाइट लेता है: 'D', 'a', 't', 'a', '\0'। नल वर्ण अंत को चिह्नित करता है, इसलिए प्रत्येक स्ट्रिंग फंक्शन '\0' मिलने तक वर्णों को स्कैन करके कार्य करता है। मानक लाइब्रेरी (string.h) strlen(), strcpy(), strcat() और strcmp() देती है, पर इन्हें स्वयं लिखने से स्ट्रिंग का वास्तविक कार्य समझ आता है, और यह प्रायोगिक परीक्षा का सामान्य प्रश्न है।

### चारों फंक्शनों का तर्क

| फंक्शन | लाइब्रेरी समतुल्य | विचार |
|---|---|---|
| my_strlen(s) | strlen | '\0' तक वर्ण गिनना |
| my_strcpy(d, s) | strcpy | '\0' सहित प्रत्येक वर्ण की प्रतिलिपि बनाना |
| my_strcat(d, s) | strcat | d के अंत तक जाकर वहाँ s की प्रतिलिपि बनाना |
| my_strcmp(a, b) | strcmp | वर्ण समान रहने तक आगे बढ़ना; पहले अंतर पर अंतर लौटाना |

© Online Student Seva · Exclusive property · Accessible only through the Pariksha Pass App · Copying, sharing or selling is strictly prohibited and will invite serious legal action.
© ऑनलाइन स्टूडेंट सेवा · विशेष संपत्ति · केवल परीक्षा पास ऐप पर उपलब्ध · कॉपी, शेयर या बिक्री पर कड़ी कानूनी कार्रवाई होगी।

### फंक्शन

```
int my_strlen(const char s[])
{
    int i = 0;
    while (s[i] != '\0')
        i++;
    return i;
}

void my_strcpy(char dest[], const char src[])
{
    int i = 0;
    while ((dest[i] = src[i]) != '\0')
        i++;
}

void my_strcat(char dest[], const char src[])
{
    int i = my_strlen(dest), j = 0;
    while ((dest[i] = src[j]) != '\0') {
        i++;
        j++;
    }
}

int my_strcmp(const char a[], const char b[])
{
    int i = 0;
    while (a[i] != '\0' && a[i] == b[i])
        i++;
    return a[i] - b[i];
}
```

my_strcpy() में असाइनमेंट शर्त के भीतर किया गया है: कॉपी किया गया वर्ण जाँचा जाता है, इसलिए '\0' की कॉपी हो जाने के बाद लूप रुकता है।

### परीक्षण प्रोग्राम

```
#include <stdio.h>

/* the four functions given above are placed here */

int main(void)
{
    char s1[30] = "Data", s2[30] = "Base", s3[30];
    printf("Length of s1   : %d\n", my_strlen(s1));
    my_strcpy(s3, s1);
    printf("Copy of s1     : %s\n", s3);
    my_strcat(s1, s2);
    printf("s1 + s2        : %s\n", s1);
    printf("strcmp(apple, apricot) = %d\n", my_strcmp("apple", "apricot"));
    printf("strcmp(ball, ball)     = %d\n", my_strcmp("ball", "ball"));
    return 0;
}
```

### आउटपुट (gcc):

```
Length of s1   : 4
Copy of s1     : Data
s1 + s2        : DataBase
strcmp(apple, apricot) = -2
strcmp(ball, ball)     = 0
```

© Online Student Seva · Exclusive property · Accessible only through the Pariksha Pass App · Copying, sharing or selling is strictly prohibited and will invite serious legal action.
© ऑनलाइन स्टूडेंट सेवा · विशेष संपत्ति · केवल परीक्षा पास ऐप पर उपलब्ध · कॉपी, शेयर या बिक्री पर कड़ी कानूनी कार्रवाई होगी।

### my_strlen() का पॉइंटर संस्करण

फंक्शन को भेजा गया ऐरे का नाम उसके पहले तत्व का पॉइंटर बन जाता है, इसलिए const char s[] और const char *s का अर्थ एक ही है। अतः पॉइंटर घटाव से लंबाई निकाली जा सकती है।

```
int ptr_strlen(const char *s)
{
    const char *p = s;
    while (*p != '\0')
        p++;
    return (int)(p - s);
}
```

पॉइंटर p नल वर्ण तक पहुँचता है, और p - s आरंभ तथा अंत के बीच वर्णों की संख्या है। "Data" के लिए यह 4 लौटाता है।

### ड्राई रन

- **my_strlen("Data"):** s[0] से s[3] तक '\0' नहीं हैं, इसलिए i = 4 हो जाता है, और s[4] '\0' है, अतः 4 लौटाया जाता है।
- **my_strcmp("apple", "apricot"):** 'a' = 'a' और 'p' = 'p', किंतु इंडेक्स 2 पर वर्ण 'p' (112) और 'r' (114) हैं। अंतर 112 - 114 = -2 ऋणात्मक है, इसलिए "apple", "apricot" से पहले आता है।

### याद रखने योग्य बातें

1. strcpy() और strcat() में गंतव्य ऐरे '\0' सहित परिणाम रखने के लिए पर्याप्त बड़ी होनी चाहिए; अन्यथा प्रोग्राम अन्य मेमोरी को अधिलेखित कर देता है।
2. मानक strcmp() केवल परिणाम के चिह्न (ऋणात्मक, शून्य या धनात्मक) की गारंटी देता है, ठीक मान की नहीं, इसलिए उत्तर -2 हमारे अपने फंक्शन पर निर्भर है।
3. स्ट्रिंग को = और == से कॉपी या तुलना नहीं किया जा सकता, क्योंकि ये ऐरे के पते पर कार्य करते हैं, उसकी सामग्री पर नहीं।

### निष्कर्ष

इस प्रकार प्रत्येक C स्ट्रिंग ऐसी char ऐरे है जिसका अंतिम प्रयुक्त तत्व '\0' है। सभी स्ट्रिंग संक्रियाएँ वर्णों पर ऐसे लूप हैं जो '\0' पर रुकते हैं, और लाइब्रेरी फंक्शन यही कार्य कुशलता से करते हैं। परीक्षा उत्तर में तर्क, कोड और संक्षिप्त ड्राई रन मिलकर पूरे अंक दिलाते हैं।

**प्रश्न 6 • इकाई 4** 10 अंक · ≈ 600–700 शब्द

**संरचनाओं की ऐरे (ऐरे ऑफ स्ट्रक्चर) क्या है? फंक्शन को स्ट्रक्चर भेजने के लिए स्ट्रक्चर के पॉइंटर का उपयोग कैसे होता है, समझाइए। ऐसा प्रोग्राम लिखिए जो तीन विद्यार्थियों के अंक संग्रहीत करे, कुल और औसत निकाले तथा टॉपर ज्ञात करे।**

### भूमिका

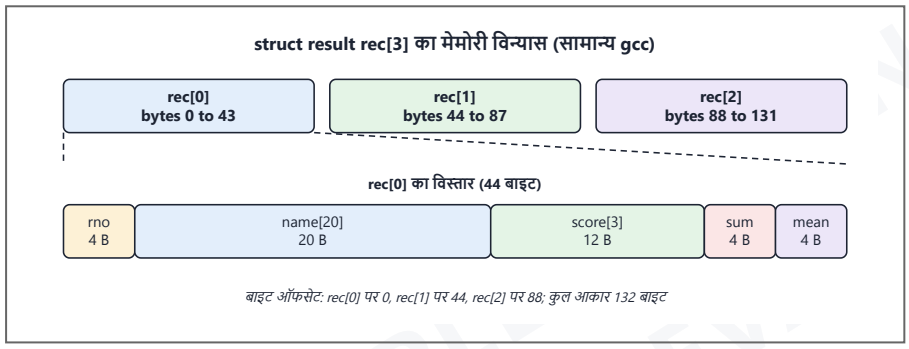
स्ट्रक्चर भिन्न प्रकार के वेरिएबलों को एक नाम के अंतर्गत समूहित करता है, जैसे विद्यार्थी का रोल नंबर, नाम और अंक। जब अनेक विद्यार्थियों के लिए वही रिकॉर्ड चाहिए तो हम **संरचनाओं की ऐरे** बनाते हैं, जिसमें प्रत्येक तत्व पूर्ण स्ट्रक्चर होता है। यह मेमोरी में छोटी तालिका या डेटाबेस बनाने का C का तरीका है।

© Online Student Seva · Exclusive property · Accessible only through the Pariksha Pass App · Copying, sharing or selling is strictly prohibited and will invite serious legal action.
© ऑनलाइन स्टूडेंट सेवा · विशेष संपत्ति · केवल परीक्षा पास ऐप पर उपलब्ध · कॉपी, शेयर या बिक्री पर कड़ी कानूनी कार्रवाई होगी।

### डिक्लेरेशन और पहुँच

```
struct result {
    int rno;
    char name[20];
    float score[3];
    float sum;
    float mean;
};
struct result rec[3];
```

- कथन struct result rec[3]; तीन स्ट्रक्चर वेरिएबल बनाता है, जो मेमोरी में एक के बाद एक संग्रहीत रहते हैं।
- सदस्य तक इंडेक्स के बाद डॉट ऑपरेटर से पहुँचते हैं: rec[1].rno, rec[1].score[2]।
- सामान्य gcc सिस्टम पर प्रत्येक स्ट्रक्चर 44 बाइट (4 + 20 + 12 + 4 + 4) लेता है, इसलिए ऐरे 132 बाइट लेती है।




*संरचनाओं की ऐरे का मेमोरी विन्यास*

### स्ट्रक्चर का पॉइंटर

पॉइंटर स्ट्रक्चर का पता रख सकता है: struct result *p = &rec[0];। तब सदस्यों तक **तीर ऑपरेटर** से पहुँचते हैं: p->rno, जो (*p).rno के समान है। फंक्शन को पॉइंटर भेजने से उसे स्ट्रक्चर के सभी बाइट कॉपी किए बिना मूल स्ट्रक्चर तक पहुँच मिलती है, जो बड़े स्ट्रक्चर के लिए तेज़ है। पैरामीटर को const घोषित करने से डेटा आकस्मिक परिवर्तन से सुरक्षित रहता है। पॉइंटर बढ़ाने, p++, से वह ऐरे के अगले स्ट्रक्चर पर चला जाता है।

© Online Student Seva · Exclusive property · Accessible only through the Pariksha Pass App · Copying, sharing or selling is strictly prohibited and will invite serious legal action.
© ऑनलाइन स्टूडेंट सेवा · विशेष संपत्ति · केवल परीक्षा पास ऐप पर उपलब्ध · कॉपी, शेयर या बिक्री पर कड़ी कानूनी कार्रवाई होगी।

### प्रोग्राम

```
#include <stdio.h>

struct result {
    int rno;
    char name[20];
    float score[3];
    float sum;
    float mean;
};

void show(const struct result *s)
{
    printf("%-4d %-8s %6.0f %7.2f\n", s->rno, s->name, s->sum, s->mean);
}

int main(void)
{
    struct result rec[3] = {
        {1, "Anil",   {78, 82, 69}},
        {2, "Bhavna", {91, 88, 94}},
        {3, "Chetan", {65, 70, 58}}
    };
    int i, j, top = 0;

    for (i = 0; i < 3; i++) {
        rec[i].sum = 0;
        for (j = 0; j < 3; j++)
            rec[i].sum += rec[i].score[j];
        rec[i].mean = rec[i].sum / 3;
        if (rec[i].sum > rec[top].sum)
            top = i;
    }
    printf("Roll Name     Total Average\n");
    for (i = 0; i < 3; i++)
        show(&rec[i]);
    printf("Topper: %s with %.0f marks\n", rec[top].name, rec[top].sum);
    return 0;
}
```

### आउटपुट (gcc):

```
Roll Name     Total Average
1    Anil        229   76.33
2    Bhavna      273   91.00
3    Chetan      193   64.33
Topper: Bhavna with 273 marks
```

### व्याख्या

1. इनिशियलाइज़र में प्रत्येक विद्यार्थी का रोल नंबर, नाम और तीन अंक दिए गए हैं।
2. बाहरी लूप प्रत्येक विद्यार्थी पर जाता है और भीतरी लूप तीनों अंक sum में जोड़ता है।
3. औसत sum / 3 है। इंडेक्स top अब तक के सर्वाधिक कुल वाले विद्यार्थी को याद रखता है।
4. फंक्शन show() पता &rec[i] प्राप्त करता है और तीर ऑपरेटर से सदस्य छापता है।
5. कुल: 78 + 82 + 69 = 229, 91 + 88 + 94 = 273 और 65 + 70 + 58 = 193, अतः भावना टॉपर है।

### सामान्य गलतियाँ

1. इंडेक्स भूल जाना: rec[0].rno के स्थान पर rec.rno लिखना।

© Online Student Seva · Exclusive property · Accessible only through the Pariksha Pass App · Copying, sharing or selling is strictly prohibited and will invite serious legal action.
© ऑनलाइन स्टूडेंट सेवा · विशेष संपत्ति · केवल परीक्षा पास ऐप पर उपलब्ध · कॉपी, शेयर या बिक्री पर कड़ी कानूनी कार्रवाई होगी।

2. पॉइंटर के साथ डॉट ऑपरेटर, या वेरिएबल के साथ तीर ऑपरेटर प्रयोग करना; दोनों कंपाइल त्रुटियाँ हैं।
3. नामों की तुलना strcmp() के बजाय == से करना।
4. printf() में float सदस्यों के लिए %d का उपयोग, जो गलत मान छापता है।

### संरचनाओं की ऐरे क्यों प्रयोग करें

एक रिकॉर्ड के सदस्य साथ रहते हैं, रिकॉर्डों को लूप से संसाधित किया जा सकता है, पूरी ऐरे को फंक्शन को भेजा या क्रमबद्ध किया जा सकता है, और वही विन्यास fwrite() से फाइल में लिखा जा सकता है।

### डॉट ऑपरेटर और तीर ऑपरेटर

| बिंदु | डॉट (.) | तीर (->) |
|---|---|---|
| किसके साथ | स्ट्रक्चर वेरिएबल | स्ट्रक्चर का पॉइंटर |
| उदाहरण | rec[0].rno | p->rno |
| समतुल्य | | (*p).rno |

### निष्कर्ष

संरचनाओं की ऐरे एक ही प्रकार के अनेक रिकॉर्ड संग्रहीत करती है, और स्ट्रक्चर का पॉइंटर फंक्शन को रिकॉर्ड कुशलता से उपयोग करने देता है। साथ मिलकर ये सरल रिकॉर्ड-संसाधन प्रोग्रामों का आधार हैं।

**प्रश्न 6 (अथवा) · इकाई 4** — 10 अंक · ≈ 600–700 शब्द

**फाइलों में रैंडम एक्सेस क्या है? fseek(), ftell(), rewind() तथा स्थिरांक SEEK_SET, SEEK_CUR और SEEK_END को समझाइए। ऐसा प्रोग्राम लिखिए जो रिकॉर्डों को बाइनरी फाइल में संग्रहीत करे और दूसरे रिकॉर्ड को उसी स्थान पर अद्यतन (अपडेट) करे।**

### भूमिका

फाइलों को दो प्रकार से संसाधित किया जा सकता है। **अनुक्रमिक एक्सेस** में डेटा आरंभ से अंत तक एक के बाद एक पढ़ा या लिखा जाता है। **रैंडम एक्सेस** में प्रोग्राम फाइल के किसी भी बाइट पर सीधे जाकर पढ़ता या लिखता है, पहले के डेटा को पढ़े बिना। जब बड़ी फाइल में एक रिकॉर्ड, जैसे एक विद्यार्थी या एक स्टॉक आइटम, पढ़ना या अद्यतन करना हो तब रैंडम एक्सेस चाहिए। C इसे **फाइल पोजीशन इंडिकेटर** से संभव बनाता है, जो अगले पठन या लेखन का स्थान चिह्नित करता है और लाइब्रेरी फंक्शनों से खिसकाया जा सकता है।

### रैंडम एक्सेस के फंक्शन

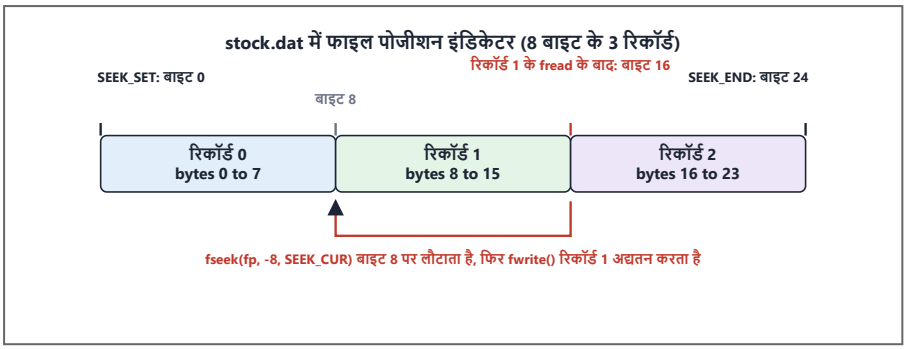
| फंक्शन | उद्देश्य |
|---|---|
| fseek(fp, offset, origin) | पोजीशन इंडिकेटर को origin से offset बाइट खिसकाता है; सफल होने पर 0 लौटाता है |
| ftell(fp) | वर्तमान स्थान, आरंभ से बाइटों में, long के रूप में लौटाता है |
| rewind(fp) | इंडिकेटर को आरंभ पर ले जाता है; fseek(fp, 0, SEEK_SET) के समान |

आर्ग्युमेंट **origin** stdio.h में परिभाषित तीन स्थिरांकों में से एक होता है:

- **SEEK_SET**: फाइल का आरंभ।
- **SEEK_CUR**: वर्तमान स्थान।

© Online Student Seva · Exclusive property · Accessible only through the Pariksha Pass App · Copying, sharing or selling is strictly prohibited and will invite serious legal action.
© ऑनलाइन स्टूडेंट सेवा · विशेष संपत्ति · केवल परीक्षा पास ऐप पर उपलब्ध · कॉपी, शेयर या बिक्री पर कड़ी कानूनी कार्रवाई होगी।

- **SEEK_END**: फाइल का अंत।




*फाइल पोजीशन इंडिकेटर और fseek() के origin*

उदाहरण: fseek(fp, 0, SEEK_END) अंत पर ले जाता है, इसलिए तब ftell(fp) फाइल का आकार देता है। fseek(fp, 2L*sizeof(struct item), SEEK_SET) तीसरे रिकॉर्ड के आरंभ पर ले जाता है, क्योंकि रिकॉर्ड गिनती 0 से आरंभ होती है। ऋणात्मक ऑफसेट, जैसे SEEK_CUR के साथ -1L*(long)sizeof(struct item), एक रिकॉर्ड पीछे ले जाता है।

मोड "wb+" बाइनरी फाइल को लिखने और पढ़ने दोनों के लिए खोलता है और उसे नए सिरे से बनाता है। बाइनरी मोड में रैंडम एक्सेस भरोसेमंद है, क्योंकि प्रत्येक रिकॉर्ड का आकार निश्चित होता है।

## प्रोग्राम

प्रोग्राम तीन रिकॉर्ड लिखता है, फाइल का आकार निकालता है, दूसरा रिकॉर्ड पढ़ता है, उसकी मात्रा में 5 जोड़ता है, उसे उसी स्थान पर वापस लिखता है और सभी रिकॉर्ड छापता है।

DEMO SAMPLE · ONLINE STUDENT SEVA

© Online Student Seva · Exclusive property · Accessible only through the Pariksha Pass App · Copying, sharing or selling is strictly prohibited and will invite serious legal action.
© ऑनलाइन स्टूडेंट सेवा · विशेष संपत्ति · केवल परीक्षा पास ऐप पर उपलब्ध · कॉपी, शेयर या बिक्री पर कड़ी कानूनी कार्रवाई होगी।

```c
#include <stdio.h>

struct item {
    int code;
    int qty;
};

int main(void)
{
    struct item w[3] = {{101, 10}, {102, 25}, {103, 40}}, r;
    FILE *fp = fopen("stock.dat", "wb+");
    if (fp == NULL) {
        printf("Cannot open file\n");
        return 1;
    }
    fwrite(w, sizeof(struct item), 3, fp);

    fseek(fp, 0, SEEK_END);
    printf("File size = %ld bytes, records = %ld\n",
           ftell(fp), ftell(fp) / (long)sizeof(struct item));

    /* update the second record in place */
    fseek(fp, 1L * sizeof(struct item), SEEK_SET);
    fread(&r, sizeof(struct item), 1, fp);
    r.qty = r.qty + 5;
    fseek(fp, -1L * (long)sizeof(struct item), SEEK_CUR);
    fwrite(&r, sizeof(struct item), 1, fp);

    rewind(fp);
    while (fread(&r, sizeof(struct item), 1, fp) == 1)
        printf("Code %d  Qty %d\n", r.code, r.qty);
    fclose(fp);
    return 0;
}
```

**आउटपुट (gcc, जहाँ int 4 बाइट का है):**

```
File size = 24 bytes, records = 3
Code 101  Qty 10
Code 102  Qty 30
Code 103  Qty 40
```

### व्याख्या

1. प्रत्येक रिकॉर्ड 8 बाइट (दो int) का है, इसलिए तीन रिकॉर्ड 24 बाइट के हैं। fseek(fp, 0, SEEK_END) के बाद ftell() 24 लौटाता है और 24 / 8 से 3 रिकॉर्ड मिलते हैं।
2. दूसरे रिकॉर्ड तक पहुँचने के लिए इंडिकेटर को आरंभ से 8 बाइट खिसकाया जाता है। fread() उसे पढ़ता है, और इंडिकेटर अब बाइट 16 पर है।
3. पढ़ने के बाद इंडिकेटर रिकॉर्ड के पीछे है, इसलिए लिखने से पहले fseek(..., -8, SEEK_CUR) से बाइट 8 पर लौटा जाता है। इस चरण के बिना नया डेटा तीसरे रिकॉर्ड को अधिलेखित कर देता।
4. rewind() आरंभ पर लौटाता है, और लूप तब तक पढ़ता है जब तक फाइल के अंत पर fread() विफल न हो जाए।

© Online Student Seva · Exclusive property · Accessible only through the Pariksha Pass App · Copying, sharing or selling is strictly prohibited and will invite serious legal action.
© ऑनलाइन स्टूडेंट सेवा · विशेष संपत्ति · केवल परीक्षा पास ऐप पर उपलब्ध · कॉपी, शेयर या बिक्री पर कड़ी कानूनी कार्रवाई होगी।

### अनुक्रमिक और रैंडम एक्सेस

| बिंदु | अनुक्रमिक | रैंडम |
|---|---|---|
| गति | आरंभ से क्रम में | सीधे किसी भी बाइट पर |
| फंक्शन | लूप में fgetc, fgets, fscanf, fread | fread, fwrite के साथ fseek, ftell, rewind |
| एक रिकॉर्ड के लिए गति | बड़ी फाइलों में धीमी | तेज़ |

### निष्कर्ष

fseek(), ftell() और rewind() फाइल पोजीशन इंडिकेटर को नियंत्रित करते हैं और रिकॉर्ड तक सीधी पहुँच संभव बनाते हैं। अद्यतन करते समय लिखने से पहले इंडिकेटर को पीछे लाना पड़ता है, क्योंकि प्रत्येक पठन या लेखन उसे आगे बढ़ा देता है।

DEMO SAMPLE · ONLINE STUDENT SEVA

© Online Student Seva · Exclusive property · Accessible only through the Pariksha Pass App · Copying, sharing or selling is strictly prohibited and will invite serious legal action.
© ऑनलाइन स्टूडेंट सेवा · विशेष संपत्ति · केवल परीक्षा पास ऐप पर उपलब्ध · कॉपी, शेयर या बिक्री पर कड़ी कानूनी कार्रवाई होगी।

## Exam Strategy परीक्षा रणनीति

1. Write every program in three parts: logic in two or three lines, the program with proper indentation, and the output; examiners give marks for each part.
2. Include the header files (#include <stdio.h>) and a return type for main(); a missing header or a missing return is a common reason for lost marks.
3. Practise a dry run (trace table) for loops and recursion: showing the value of each variable step by step proves that you understand the logic.
4. Draw a neat flowchart when the question says so, using the standard symbols: oval, parallelogram, rectangle and diamond.
5. Know the difference pairs: while and do-while, call by value and call by reference, structure and union, array and pointer.
6. Practise one long program answer in 25 minutes and one short answer in 6 minutes.
7. Do not depend on this demo or any single paper: finish the full syllabus from the prescribed books and practise on a real compiler.

1. हर प्रोग्राम तीन भागों में लिखें: दो-तीन पंक्तियों में तर्क, सही इंडेंटेशन के साथ प्रोग्राम और आउटपुट; परीक्षक प्रत्येक भाग पर अंक देते हैं।
2. हेडर फाइलें (#include <stdio.h>) और main() का रिटर्न टाइप अवश्य लिखें; हेडर या return छूटना अंक कटने का सामान्य कारण है।
3. लूप और रिकर्शन के लिए ड्राई रन (ट्रेस टेबल) का अभ्यास करें: प्रत्येक वेरिएबल का मान चरण-दर-चरण दिखाना आपकी समझ प्रमाणित करता है।
4. जब प्रश्न में कहा जाए तब मानक चिह्नों (अंडाकार, समानांतर चतुर्भुज, आयत और हीरा) से साफ फ्लोचार्ट बनाएँ।
5. जोड़ी वाले अंतर याद रखें: while और do-while, कॉल बाय वैल्यू और कॉल बाय रेफरेंस, स्ट्रक्चर और यूनियन, ऐरे और पॉइंटर।
6. एक दीर्घ प्रोग्राम उत्तर 25 मिनट में और एक लघु उत्तर 6 मिनट में लिखने का अभ्यास करें।
7. इस डेमो या किसी एक पेपर पर निर्भर न रहें: निर्धारित पुस्तकों से पूरा पाठ्यक्रम पूरा करें और असली कंपाइलर पर अभ्यास करें।

**Final reminder / अंतिम सलाह:** Even though this paper covers high-probability topics, do not depend on it fully — revise the entire syllabus. इस पर पूरी तरह निर्भर न रहें; पूरा पाठ्यक्रम पढ़ें।
Online Student Seva · Pariksha Pass App · Edition Oct 2026

© Online Student Seva · Exclusive property · Accessible only through the Pariksha Pass App · Copying, sharing or selling is strictly prohibited and will invite serious legal action.
© ऑनलाइन स्टूडेंट सेवा · विशेष संपत्ति · केवल परीक्षा पास ऐप पर उपलब्ध · कॉपी, शेयर या बिक्री पर कड़ी कानूनी कार्रवाई होगी।

## Like the answer quality? उत्तरों की गुणवत्ता पसंद आई?

# Unlock the full sets in the Pariksha Pass App

This Demo Set shows the real paper format and answer quality for **Principles of Programming using C**. The complete Guessing Paper for your university gives you, for this subject:

- ✓ **3 full sets** – Set A (Most Probable), Set B (Highly Likely) and Set C (Alternate)
- ✓ **30 objective, 15 short-answer and 24 long-answer questions** with model answers, in English and in Hindi
- ✓ Every unit of the syllabus covered, with labelled diagrams, tables and worked examples
- ✓ An exam-style question paper with your university's header, a syllabus coverage table and a book list
- ✓ An "Asked in" tag wherever a question matches a past university paper

**पूर्ण गेसिंग पेपर में :** इसी विषय के 3 पूर्ण सेट (A, B, C), 30 वस्तुनिष्ठ, 15 लघु उत्तर तथा 24 दीर्घ उत्तर वाले प्रश्न — सभी आदर्श उत्तरों सहित, अंग्रेज़ी व हिन्दी दोनों में; पूरे पाठ्यक्रम की सभी इकाइयाँ, चित्र, तालिकाएँ और हल किए गए उदाहरण।

**Available only in the Pariksha Pass App.** Open the app, choose your university, course, semester and subject, and unlock your paper. केवल परीक्षा पास ऐप पर उपलब्ध — ऐप खोलें, अपना विश्वविद्यालय, कोर्स, सेमेस्टर और विषय चुनें और अपना पेपर अनलॉक करें।

**Please note / ध्यान दें:** The questions in this Demo Set are **not** the questions of the actual Set A, B or C; they are sample questions from the same syllabus, written to show the paper format and the answer quality. इस डेमो सेट के प्रश्न वास्तविक सेट A, B, C के प्रश्न नहीं हैं; ये उसी पाठ्यक्रम के नमूना प्रश्न हैं, जो केवल पेपर का प्रारूप व उत्तरों की गुणवत्ता दिखाने के लिए बनाए गए हैं।

Our sets focus on high-probability topics, but no paper can promise what will be asked — use them for practice along with your text book. हमारे सेट अधिक संभावित टॉपिक पर केंद्रित हैं, पर कोई भी पेपर यह वादा नहीं कर सकता कि क्या पूछा जाएगा — पाठ्यपुस्तक के साथ अभ्यास के लिए उपयोग करें।

Online Student Seva · Pariksha Pass App · Demo edition Oct 2026

**Download Pariksha Pass App >**

Get all 3 full Guessing Paper sets on Google Play

© Online Student Seva · Exclusive property · Accessible only through the Pariksha Pass App · Copying, sharing or selling is strictly prohibited and will invite serious legal action.
© ऑनलाइन स्टूडेंट सेवा · विशेष संपत्ति · केवल परीक्षा पास ऐप पर उपलब्ध · कॉपी, शेयर या बिक्री पर कड़ी कानूनी कार्रवाई होगी।