

C foundations learning review

C बुनियाद पुनराभ्यास

12 questions · 12 marks · Free · Untimed

12 प्रश्न · 12 अंक · निःशुल्क · समय सीमा नहीं

Correct +1. Wrong or unattempted 0. No negative marks, partial credit or pass cutoff. MSQ answers require the exact correct set.

सही उत्तर +1। गलत या अनुत्तरित प्रश्न 0। ऋणात्मक अंक, आंशिक अंक या उत्तीर्ण होने की सीमा नहीं। MSQ में सभी और केवल सही विकल्प चुनें।

Attempt the questions first, then compare your trace with the full solutions. This is learning feedback for the five foundation lessons.

पहले प्रश्न हल करें, फिर अपने ट्रेस की पूरे हल से तुलना करें। यह पाँच बुनियादी पाठों को सीखने के लिए प्रतिपुष्टि है।

Original bilingual study material. Explicit C11 teaching convention. This is a bounded foundation module, not full C or full GATE CS coverage. Hindi supports learning; GATE examination content is in English.

मौलिक द्विभाषी अध्ययन सामग्री। शिक्षण के लिए C11 स्पष्ट रूप से चुना गया है। यह सीमित बुनियादी मॉड्यूल है, पूरा C या पूरा GATE CS पाठ्यक्रम नहीं। हिंदी सीखने में सहायता करती है; GATE परीक्षा की सामग्री अंग्रेज़ी में होती है।

Contents

विषय सूची

Choose English or Hindi. Questions come before the answer key and full solutions.

अंग्रेज़ी या हिंदी चुनें। प्रश्नों के बाद उत्तर और पूरे हल दिए गए हैं।

C foundation learning review · English.....	3
C बुनियाद पुनराभ्यास · हिंदी	16

Internal links and PDF bookmarks support navigation. Code blocks are independent examples.

अंदरूनी लिंक और PDF बुकमार्क से आगे जाएँ। हर कोड ब्लॉक स्वतंत्र उदाहरण है।

C foundation learning review

Trace each program using its current integer values. For a call, keep the caller's variables separate from the helper's parameters and locals. For a loop, include the test that finally fails. Write a short state trace alongside your answer so that you can check the reasoning afterward.

This is a free, untimed review of the five C foundation lessons. It has 12 questions worth 1 mark each, for 12 marks in total. A correct answer earns +1; a wrong or unattempted answer earns 0. There are no negative or partial marks and no pass threshold. It covers this bounded C foundation, not the full GATE CS syllabus or a full-length GATE paper.

- MCQ: choose exactly one option.
- MSQ: choose every correct option. The selected set must match the answer exactly; a missing correct option or an extra incorrect option earns 0. The order in which you select options does not matter.
- NAT: enter the requested integer. These three questions have exact integer answers.
- Check the solutions after attempting the questions. Revisit the relevant state trace and retry without a timer; the score is learning feedback, not a qualification prediction.

Use C11 for all questions. Every displayed code block is a complete, independent program: start again with its own initializers. All computed values here fit the portable signed `int` range -32767 through 32767. Every evaluated division has a nonzero divisor and a representable quotient. Calls have typed definitions before use, and each helper returns an integer on every reached path. No particular `int` width is assumed. The values printed by `%d` are integers, and `\n` ends the output line.

Questions

Question 1 · MCQ · 1 mark

What pair is printed? Choose exactly one option.

C code 1

```
#include <stdio.h>

int main(void) {
    int a=5,b=8;
    a=b-a;
    b=b+a;
    printf("%d %d\n",a,b);
    return 0;
}
```

- A. 3 11
- B. 3 8
- C. 5 13
- D. 3 3

Question 2 · NAT · 1 mark

Enter the integer printed.

C code 2

```
#include <stdio.h>

int main(void) {
    int total=58,group=9;
    int leftover=total-group*(total/group);
    printf("%d\n",leftover);
    return 0;
}
```

Question 3 · MCQ · 1 mark

What triple is printed in the order n, left, right? Choose exactly one option.

C code 3

```
#include <stdio.h>

int main(void) {
    int n=2;
    int left=n++;
    int right=++n;
    printf("%d %d %d\n",n,left,right);
    return 0;
}
```

- A. 4 3 4
- B. 3 2 3
- C. 4 2 4
- D. 4 2 3

Question 4 · MSQ · 1 mark

For int p=-17 and q=5, select every true statement.

C code 4

```
#include <stdio.h>

int main(void) {
    int p=-17,q=5;
    printf("%d %d %d\n",p/q,p%q,(p/q)*q+p%q);
    return 0;
}
```

- A. p/q equals -3
- B. p%q equals 3
- C. (p/q)*q+p%q equals p
- D. p/q equals -4

Question 5 · MCQ · 1 mark

What value of tag is printed? Choose exactly one option.

C code 5

```
#include <stdio.h>

int main(void) {
    int c=7,tag=0;
    if(c<5){tag=1;}else if(c%2==1){tag=2;}else{tag=3;}
    printf("%d\n",tag);
    return 0;
}
```

- A. 1
- B. 2
- C. 3
- D. 0

Question 6 · MSQ · 1 mark

For int a=0 and b=6, which expressions have value 1? Select every correct option.

C code 6

```
#include <stdio.h>

int main(void) {
    int a=0,b=6;
    printf("%d %d %d %d\n",!a,a||(b>4),a&&(b>0),!(b==6));
    return 0;
}
```

- A. !a
- B. a || (b > 4)
- C. a && (b > 0)
- D. !(b == 6)

Question 7 · NAT · 1 mark

Enter the number of body executions.

C code 7

```
#include <stdio.h>

int main(void) {
    int count=0;
    for(int k=3;k<=12;k+=3){count++;}
    printf("%d\n",count);
    return 0;
}
```

Question 8 · MCQ · 1 mark

What integer is printed? Trace each division and remainder. Choose exactly one option.

C code 8

```
#include <stdio.h>

int main(void) {
    int n=19, sum=0;
    while(n>0){sum+=n%3;n/=3;}
    printf("%d\n", sum);
    return 0;
}
```

- A. 4
- B. 6
- C. 19
- D. 3

Question 9 · MSQ · 1 mark

Select every true statement about this for loop. Count the final failed test.

C code 9

```
#include <stdio.h>

int main(void) {
    int sum=0, i;
    for(i=0; i<4; i++){sum+=2*i;}
    printf("%d %d\n", sum, i);
    return 0;
}
```

- A. The body runs 4 times
- B. The condition is tested 4 times
- C. The final sum is 12
- D. The update i++ runs 4 times

Question 10 · MCQ · 1 mark

What pair is printed? Choose exactly one option.

C code 10

```
#include <stdio.h>

int bump(int x){x+=3;return x;}

int main(void) {
    int v=4;
    int result=bump(v);
    printf("%d %d\n",v,result);
    return 0;
}
```

- A. 7 7
- B. 4 7
- C. 4 4
- D. 7 4

Question 11 · NAT · 1 mark

Enter the final printed integer.

C code 11

```
#include <stdio.h>

int accumulate(int n){int s=1;for(int k=1;k<=n;k++){s+=2*k;}return s;}

int main(void) {
    int r=accumulate(3);
    int t=accumulate(1);
    printf("%d\n",r-t);
    return 0;
}
```

Question 12 · MSQ · 1 mark

Use only inputs -3 through 3. Select every true statement about choose.

C code 12

```
#include <stdio.h>

int choose(int n){if(n<0){return -n;}return n+1;}

int main(void) {
    int a=choose(-3);
    int b=choose(0);
    int c=choose(2);
    int value=-2;
    int result=choose(value);
    printf("%d %d %d %d %d\n",a,b,c,value,result);
    return 0;
}
```

- A. choose(-3) returns 3
- B. choose(0) returns 0
- C. choose(2) returns 3
- D. Calling choose(value) does not change the caller's int variable value

Answers and full solutions

Use the reasoning below to compare states, not just final numbers.

Question 1

Answer: A

1. Initially $a = 5$, $b = 8$.
2. $a = b - a$ reads the current 8 and 5, computes 3, and stores it in a. State: $a = 3$, $b = 8$.
3. $b = b + a$ now reads 8 and the updated 3, so $b = 11$.
4. The output order is a, b, giving 3 11.

Option checks

- A: Correct. Both assignments are applied in order.
- B: Incorrect. `3 8` stops after the first assignment and ignores the update to b.
- C: Incorrect. `5 13` retains the old $a = 5$ and uses it in the second assignment, although a has already become 3.
- D: Incorrect. $b = b + a$ adds 8 and 3; it does not copy a into b.

Question 2

Answer: 4

1. Both operands of `total / group` are integers: $58 / 9$ gives 6 whole groups.
2. Multiply the group size by that quotient: $9 * 6 = 54$.
3. Subtract the grouped items from the original total: $58 - 54 = 4$.
4. Thus `leftover = 4`, and the required integer is 4. Check: $6 * 9 + 4 = 58$, with both $0 \leq 4$ and $4 < 9$.

6 is the quotient, not the leftover. 54 is the number placed into complete groups. 0 would result from treating the division as an exact real-number quotient; C integer division has already discarded the fractional part before multiplication.

Question 3

Answer: C

1. Start with $n = 2$.
2. $\text{left} = n++$ uses the old value 2 as the expression value, so $\text{left} = 2$; by the end of this statement $n = 3$.
3. $\text{right} = ++n$ first increases n from 3 to 4, then supplies 4, so $\text{right} = 4$.
4. The separate statements are complete before `printf`. The requested order n , left , right gives 4 2 4.

Option checks

- A: Incorrect. $\text{left} = 3$ treats postfix $n++$ as if it supplied the incremented value.
- B: Incorrect. There are two executed increments, so n does not stop at 3; right receives 4.
- C: Correct. The saved values are 2 and 4, while the final n is 4.
- D: Incorrect. $\text{right} = 3$ treats prefix $++n$ as if it supplied the old value.

Question 4

Answer: A, C

1. The mathematical quotient is -3.4. C11 integer division discards the fractional part toward zero, giving $p / q = -3$.
2. Use the reconstruction identity: $-17 = (-3) * 5 + r$, so $r = -2$. Thus $p \% q = -2$.
3. Substituting the two C results gives $(p / q) * q + p \% q = (-3) * 5 + (-2) = -17$, which equals p .
4. The selected set must therefore be exactly A and C.

Option checks

- A: True. Truncating -3.4 toward zero gives -3.
- B: False. A positive remainder 3 would go with quotient -4, but that is not C11's quotient here. The C remainder is -2.
- C: True. The divisor is nonzero and the quotient is representable, so the stated identity applies and gives -17.
- D: False. -4 rounds downward toward negative infinity; C11 integer division truncates toward zero.

Question 5

Answer: B

1. Initially $c = 7$, $tag = 0$.
2. The first test $c < 5$ is false, so $tag = 1$ is skipped.
3. Next, $c \% 2$ is 1, so $c \% 2 == 1$ is true. Execute $tag = 2$.
4. The final `else` is skipped because the preceding branch was selected. The output is 2.

Option checks

- A: Incorrect. The assignment of 1 is in a branch whose test is false.
- B: Correct. The odd-number test is the selected branch.
- C: Incorrect. The final `else` does not execute after a true `else if` branch.
- D: Incorrect. The initial 0 is replaced by the executed $tag = 2$ assignment.

Question 6

Answer: A, B

1. a is 0, so it is false in a logical test. b is 6.
2. $!a$ is $!0$, which is 1.
3. For $a || (b > 4)$, the left operand is false, so the right operand is evaluated. $6 > 4$ is true, and logical OR returns 1.
4. For $a \&\& (b > 0)$, the left operand is false, so the right operand is skipped and logical AND returns 0.
5. For $!(b == 6)$, the equality is true and has value 1; negating it gives 0. Exactly A and B have value 1.

Option checks

- A: True. Logical negation turns zero into 1.
- B: True. At least one operand is true, so $||$ returns 1, not the variable b 's value 6.
- C: False. A false left operand makes $\&\&$ return 0 without evaluating the right operand.
- D: False. The equality $b == 6$ is true, so its negation is 0.

Question 7

Answer: 4

1. Initialization sets $\text{count} = 0$ and the loop's $k = 3$.
2. At $k = 3$, the test is true: count becomes 1; update gives $k = 6$.
3. At $k = 6$, the test is true: count becomes 2; update gives $k = 9$.
4. At $k = 9$, the test is true: count becomes 3; update gives $k = 12$.
5. At $k = 12$, the test is still true because the bound is inclusive: count becomes 4; update gives $k = 15$.
6. At $k = 15$, the test is false. Four bodies and four updates have executed, with five condition tests. The requested body count is 4.

3 omits the included endpoint 12. 5 counts the final failed condition as if a body ran there. 10 counts every integer from 3 to 12 instead of the visited values separated by 3.

Question 8

Answer: D

1. Initially $n = 19$, $\text{sum} = 0$; $19 > 0$ is true.
2. First body: $19 \% 3 = 1$, so $\text{sum} = 1$; then $n /= 3$ stores $19 / 3 = 6$.
3. Second test is $6 > 0$, true. $6 \% 3 = 0$, so sum stays 1; integer division sets $n = 2$.
4. Third test is $2 > 0$, true. $2 \% 3 = 2$, so $\text{sum} = 3$; integer division sets $n = 0$.
5. The test $0 > 0$ is false, so the loop ends. Three bodies and four tests occurred; the printed quantity is the remainder sum 3.

Option checks

- A: Incorrect. 4 is the number of condition tests, not the sum of the remainders. The final failed test adds nothing.
- B: Incorrect. 6 is the first updated value of n , not the final value printed from sum .
- C: Incorrect. The starting 19 is not added to sum ; only $n \% 3$ is added in each body.
- D: Correct. The added remainders are 1, 0 and 2, totaling 3.

Question 9

Answer: A, C, D

1. Start with $\text{sum} = 0$, then initialize $i = 0$.
2. At $i = 0$, the test $0 < 4$ is true. Add $2 * 0 = 0$, keeping $\text{sum} = 0$; update to $i = 1$.
3. At $i = 1$, the test $1 < 4$ is true. Add 2, so $\text{sum} = 2$; update to $i = 2$.
4. At $i = 2$, the test $2 < 4$ is true. Add 4, so $\text{sum} = 6$; update to $i = 3$.
5. At $i = 3$, the test $3 < 4$ is true. Add 6, so $\text{sum} = 12$; update to $i = 4$.
6. At $i = 4$, the test $4 < 4$ is false. There is no fifth body or fifth update. Body count is 4, update count is 4, test count is 5. The output is 12 4; the true statements are A, C and D.

Option checks

- A: True. The body runs at $i = 0, 1, 2, 3$.
- B: False. There are four successful tests and one final false test, making five.
- C: True. The accumulator contains $0 + 2 + 4 + 6 = 12$.
- D: True. The update follows each of the four completed bodies, including the one that changes i from 3 to 4.

Question 10

Answer: B

1. In the caller, $v = 4$. The argument expression in `bump(v)` supplies 4.
2. The helper's separate parameter starts as $x = 4$. $x += 3$ makes the helper's $x = 7$.
3. `return x` ends the helper and returns 7. The caller stores this in `result`.
4. Caller v was never assigned a new value; it stays 4. Output order is v, result , so the pair is 4 7.

Option checks

- A: Incorrect. 7 7 assumes the helper's parameter update changes caller v .
- B: Correct. The argument is copied and the returned 7 is stored in `result`.
- C: Incorrect. 4 4 preserves the caller correctly but ignores the helper's addition before its return.
- D: Incorrect. It both changes caller v without an assignment and uses the old value as the return result.

Question 11

Answer: 10

1. First call: parameter $n = 3$; local s starts at 1, as its initializer says.
2. For $k = 1, 2, 3$, add 2, 4, 6. The successive s values are 3, 7, 13. At $k = 4$ the test fails, so return 13 and store $r = 13$.
3. Second call: parameter $n = 1$; this call's new local s starts at 1 again.
4. At $k = 1$, add 2, so $s = 3$. Update to $k = 2$; the test fails. Return 3 and store $t = 3$.
5. The caller prints $r - t = 13 - 3 = 10$. The required integer is 10.

13 stops after the first call and ignores the printed subtraction. -2 would result from incorrectly continuing the second s from 13, obtaining 15 and subtracting it from 13. 12 subtracts the second call's initial 1 rather than its returned 3. A numeric answer alone does not demonstrate that both local initializations were traced.

Question 12

Answer: A, C, D

1. For every supplied integer from -3 through 3, check $n < 0$. A negative input takes return $-n$; zero and positive inputs take return $n + 1$. The negative return ends its call, so the later return is not also executed.
2. `choose(-3)` takes the negative branch and returns 3.
3. `choose(0)` has a false negative test and returns $0 + 1 = 1$.
4. `choose(2)` also takes the final return and gives $2 + 1 = 3$.
5. For caller `value = -2`, the helper receives a separate parameter -2 and returns 2. Caller `value` stays -2; `result` becomes 2. Thus the displayed program prints 3 1 3 -2 2.
6. The true statements are exactly A, C and D. Across the stated input range, outputs for -3, -2, -1, 0, 1, 2, 3 are 3, 2, 1, 1, 2, 3, 4; all arithmetic is within the stated small-integer range.

Option checks

- A: True. For -3, $n < 0$ is true and $-n$ equals 3.
- B: False. Zero is not negative; it reaches return $n + 1$ and returns 1.
- C: True. Positive 2 follows the nonnegative route and returns 3.
- D: True. The helper receives the argument's integer value; returning a calculation does not assign to caller `value`. This remains true throughout the stated input range.

What to revisit

Question 1-Question 2: current state and integer arithmetic. Question 3-Question 4: expression values and signed division. Question 5-Question 6: chosen branches and logical tests. Question 7-Question 9: exact loop states and counts. Question 10-Question 12: parameters, local initializers and returns. For a missed item, correct the first state where your trace diverged, then work forward again.

Source note

C11 semantics are referenced to the [WG14 N1570 committee draft](#). For assignments, arithmetic, logical operators, loops and functions, see §§6.5, 6.8.4-6.8.6 and 6.9.1.

Analogy and its limits

Treat a trace as a ledger with a line for each change. An assignment revises one entry, a branch chooses which lines are used, and a loop repeats a group of lines until its test fails. A function call opens a separate worksheet for its copied parameters and local variables, then sends back one result. Comparing ledgers reveals where two apparently plausible answers start to differ.

Quick reference

- 12 questions, 12 marks; free and untimed; +1 correct, 0 wrong or blank.
- MCQ: one choice. MSQ: exact correct set, no partial marks. NAT: exact integer.
- No negative marks or pass threshold. Use explanations to identify what to practise next.
- Start every program from its own initializers; previous questions do not change its state.
- Assignment uses the current state. Separate increment statements complete before the next statement.
- C11 signed integer division truncates toward zero; remainder reconstructs the dividend.
- `&&` and `||` short-circuit. Logical results are 0 or 1.
- Count successful bodies separately from condition tests and updates.
- An integer parameter is separate from the caller's variable. Each shown local initializer runs anew on its call.
- Read the `printf` arguments in the stated output order; do not confuse that order with a general evaluation-order guarantee.

[Back to contents](#)

C बुनियाद पुनराभ्यास

हर प्रोग्राम का ट्रेस उसके वर्तमान पूर्णांक मानों से करें। कॉल में caller के चरों को helper के पैरामीटर और स्थानीय चरों से अलग रखें। लूप में अंत में गलत होने वाली जाँच भी लिखें। उत्तर के साथ अवस्था का छोटा ट्रेस बनाएँ, ताकि बाद में तर्क जाँच सकें।

यह पाँच C बुनियादी पाठों का निःशुल्क, बिना समय-सीमा वाला पुनराभ्यास है। इसमें 12 प्रश्न हैं; प्रत्येक 1 अंक का है, कुल 12 अंक। सही उत्तर पर +1 और गलत या छोड़े गए उत्तर पर 0 मिलता है। न ऋणात्मक अंक हैं, न आंशिक अंक, न उत्तीर्ण होने की कोई सीमा। यह इसी सीमित C बुनियाद का अभ्यास है, पूरा GATE CS पाठ्यक्रम या पूर्ण GATE प्रश्नपत्र नहीं।

- MCQ: ठीक एक विकल्प चुनें।
- MSQ: सभी सही विकल्प चुनें। चुना गया समुच्चय उत्तर से पूरी तरह मिलना चाहिए; कोई सही विकल्प छूटने या अतिरिक्त गलत विकल्प चुनने पर 0 मिलेगा। विकल्प चुनने का क्रम मायने नहीं रखता।
- NAT: माँगा गया पूर्णांक लिखें। इन तीन प्रश्नों के उत्तर सटीक पूर्णांक हैं।
- प्रश्न आजमाने के बाद हल देखें। संबंधित अवस्था-ट्रेस फिर पढ़ें और बिना टाइमर दोबारा प्रयास करें; अंक सीखने की प्रतिक्रिया हैं, चयन या योग्यता की भविष्यवाणी नहीं।

सभी प्रश्नों में C11 उपयोग करें। हर दिखाया code block पूरा और स्वतंत्र प्रोग्राम है: उसके अपने शुरुआती मानों से फिर शुरू करें। यहाँ गणना के सभी मान पोर्टेबल signed int सीमा -32767 से 32767 में आते हैं। हर मूल्यांकित भाग में भाजक गैर-शून्य है और भागफल निरूपित हो सकता है। कॉल से पहले प्रकार सहित फ़ंक्शन-परिभाषा दी गई है और हर पहुँचने वाले रास्ते पर helper पूर्णांक लौटाता है। int की कोई खास बिट-चौड़ाई नहीं मानी गई है। %d से छपने वाले मान पूर्णांक हैं और \n आउटपुट पंक्ति समाप्त करता है।

प्रश्न

प्रश्न 1 • MCQ • 1 अंक

कौन-सी जोड़ी छपती है? ठीक एक विकल्प चुनिए।

C कोड 1

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

```
int main(void) {
    int a=5,b=8;
    a=b-a;
    b=b+a;
    printf("%d %d\n",a,b);
    return 0;
}
```

- A. 3 11
- B. 3 8
- C. 5 13
- D. 3 3

प्रश्न 2 • NAT • 1 अंक

छपने वाला पूर्णांक लिखिए।

C कोड 2

```
#include <stdio.h>

int main(void) {
    int total=58,group=9;
    int leftover=total-group*(total/group);
    printf("%d\n",leftover);
    return 0;
}
```

प्रश्न 3 • MCQ • 1 अंक

n, left, right के क्रम में कौन-से तीन मान छपते हैं? ठीक एक विकल्प चुनिए।

C कोड 3

```
#include <stdio.h>

int main(void) {
    int n=2;
    int left=n++;
    int right=++n;
    printf("%d %d %d\n",n,left,right);
    return 0;
}
```

- A. 4 3 4
- B. 3 2 3
- C. 4 2 4
- D. 4 2 3

प्रश्न 4 • MSQ • 1 अंक

int p=-17 और q=5 पर सभी सही कथन चुनिए।

C कोड 4

```
#include <stdio.h>

int main(void) {
    int p=-17,q=5;
    printf("%d %d %d\n",p/q,p%q,(p/q)*q+p%q);
    return 0;
}
```

- A. p/q का मान -3 है
- B. p%q का मान 3 है
- C. (p/q)*q+p%q का मान p है
- D. p/q का मान -4 है

प्रश्न 5 • MCQ • 1 अंक

tag का कौन-सा मान छपता है? ठीक एक विकल्प चुनिए।

C कोड 5

```
#include <stdio.h>

int main(void) {
    int c=7,tag=0;
    if(c<5){tag=1;}else if(c%2==1){tag=2;}else{tag=3;}
    printf("%d\n",tag);
    return 0;
}
```

- A. 1
- B. 2
- C. 3
- D. 0

प्रश्न 6 • MSQ • 1 अंक

int a=0 और b=6 पर किन व्यंजकों का मान 1 है? सभी सही विकल्प चुनिए।

C कोड 6

```
#include <stdio.h>

int main(void) {
    int a=0,b=6;
    printf("%d %d %d %d\n", !a,a || (b>4), a&&(b>0), !(b==6));
    return 0;
}
```

- A. !a
- B. a || (b > 4)
- C. a && (b > 0)
- D. !(b == 6)

प्रश्न 7 • NAT • 1 अंक

body के चलने की संख्या लिखिए।

C कोड 7

```
#include <stdio.h>

int main(void) {
    int count=0;
    for(int k=3;k<=12;k+=3){count++;}
    printf("%d\n",count);
    return 0;
}
```

प्रश्न 8 • MCQ • 1 अंक

कौन-सा पूर्णांक छपता है? हर भाग और शेषफल का ट्रेस कीजिए। ठीक एक विकल्प चुनिए।

C कोड 8

```
#include <stdio.h>

int main(void) {
    int n=19, sum=0;
    while(n>0){sum+=n%3; n/=3;}
    printf("%d\n", sum);
    return 0;
}
```

- A. 4
- B. 6
- C. 19
- D. 3

प्रश्न 9 • MSQ • 1 अंक

इस for लूप के सभी सही कथन चुनिए। अंतिम असफल जाँच भी गिनिए।

C कोड 9

```
#include <stdio.h>

int main(void) {
    int sum=0, i;
    for(i=0; i<4; i++){sum+=2*i;}
    printf("%d %d\n", sum, i);
    return 0;
}
```

- A. body 4 बार चलती है
- B. शर्त 4 बार जाँची जाती है
- C. अंतिम sum 12 है
- D. अपडेट i++ 4 बार चलता है

प्रश्न 10 • MCQ • 1 अंक

कौन-सी जोड़ी छपती है? ठीक एक विकल्प चुनिए।

C कोड 10

```
#include <stdio.h>

int bump(int x){x+=3;return x;}

int main(void) {
    int v=4;
    int result=bump(v);
    printf("%d %d\n",v,result);
    return 0;
}
```

- A. 7 7
- B. 4 7
- C. 4 4
- D. 7 4

प्रश्न 11 • NAT • 1 अंक

अंत में छपने वाला पूर्णांक लिखिए।

C कोड 11

```
#include <stdio.h>

int accumulate(int n){int s=1;for(int k=1;k<=n;k++){s+=2*k;}return s;}

int main(void) {
    int r=accumulate(3);
    int t=accumulate(1);
    printf("%d\n",r-t);
    return 0;
}
```

प्रश्न 12 • MSQ • 1 अंक

केवल -3 से 3 तक के input लें। choose के सभी सही कथन चुनिए।

C कोड 12

```
#include <stdio.h>

int choose(int n){if(n<0){return -n;}return n+1;}

int main(void) {
    int a=choose(-3);
    int b=choose(0);
    int c=choose(2);
    int value=-2;
    int result=choose(value);
    printf("%d %d %d %d %d\n",a,b,c,value,result);
    return 0;
}
```

- A. choose(-3) का लौटाया मान 3 है
- B. choose(0) का लौटाया मान 0 है
- C. choose(2) का लौटाया मान 3 है
- D. choose(value) की कॉल caller के int चर value को नहीं बदलती

उत्तर और पूरे हल

नीचे दिए तर्क से केवल अंतिम संख्याएँ नहीं, अवस्थाएँ भी मिलाएँ।

प्रश्न 1

उत्तर: A

1. शुरू में $a = 5$, $b = 8$ हैं।
2. $a = b - a$ वर्तमान 8 और 5 पढ़कर 3 निकालता है और उसे a में रखता है। अवस्था: $a = 3$, $b = 8$ ।
3. $b = b + a$ अब 8 और अपडेट हुआ 3 पढ़ता है, इसलिए $b = 11$ ।
4. आउटपुट क्रम a , b है, इसलिए 3 11 छपता है।

विकल्पों की जाँच

- A: सही। दोनों असाइनमेंट क्रम से लागू किए गए हैं।
- B: गलत। 3 8 पहली असाइनमेंट के बाद रुक जाता है और b का अपडेट छोड़ता है।
- C: गलत। 5 13 पुराने $a = 5$ को बनाए रखता है और दूसरी असाइनमेंट में उपयोग करता है, जबकि a पहले ही 3 हो चुका है।
- D: गलत। $b = b + a$ में 8 और 3 जुड़ते हैं; यह a को b में कॉपी नहीं करता।

प्रश्न 2

उत्तर: 4

1. total / group के दोनों operands पूर्णांक हैं: $58 / 9$ से 6 पूरे समूह मिलते हैं।
2. समूह के आकार को भागफल से गुणा करें: $9 * 6 = 54$ ।
3. कुल में से समूहों में लगी संख्या घटाएँ: $58 - 54 = 4$ ।
4. अतः leftover = 4 और उत्तर पूर्णांक 4 है। जाँच: $6 * 9 + 4 = 58$, जहाँ $0 \leq 4$ और $4 < 9$ दोनों सही हैं।

6 भागफल है, बची मात्रा नहीं। 54 पूरे समूहों में रखी संख्या है। 0 तब आएगा जब भाग को वास्तविक संख्याओं वाला सटीक भागफल माना जाए; C का पूर्णांक भाग गुणा से पहले ही भिन्न भाग हटा चुका है।

प्रश्न 3

उत्तर: C

1. शुरू में $n = 2$ है।
2. $left = n++$ में व्यंजक का मान पुराना 2 है, इसलिए $left = 2$; इस कथन के अंत तक $n = 3$ हो जाता है।
3. $right = ++n$ पहले n को 3 से 4 करता है, फिर 4 देता है, इसलिए $right = 4$ ।
4. `printf` से पहले दोनों अलग कथन पूरे हो चुके हैं। माँगे गए n , $left$, $right$ क्रम में 4 2 4 छपता है।

विकल्पों की जाँच

- A: गलत। $left = 3$ मानने पर postfix $n++$ को बढ़ा हुआ मान देने वाला समझ लिया गया है।
- B: गलत। दो बढ़ोतरी वास्तव में चलती हैं, इसलिए n का अंतिम मान 3 नहीं है; $right$ को 4 मिलता है।
- C: सही। रखे गए मान 2 और 4 हैं, जबकि अंतिम n का मान 4 है।
- D: गलत। $right = 3$ मानने पर prefix $++n$ को पुराना मान देने वाला समझ लिया गया है।

प्रश्न 4

उत्तर: A, C

1. गणितीय भागफल -3.4 है। C11 का पूर्णांक भाग भिन्न भाग हटाकर शून्य की ओर जाता है, इसलिए $p / q = -3$ ।
2. पुनर्निर्माण पहचान उपयोग करें: $-17 = (-3) * 5 + r$, इसलिए $r = -2$ । अतः $p \% q = -2$ ।
3. C के दोनों परिणाम रखने पर $(p / q) * q + p \% q = (-3) * 5 + (-2) = -17$, जो p के बराबर है।
4. इसलिए चुना गया समुच्चय ठीक A और C होना चाहिए।

विकल्पों की जाँच

- A: सही। -3.4 को शून्य की ओर काटने पर -3 मिलता है।
- B: गलत। धनात्मक शेषफल 3 भागफल -4 के साथ मेल खाता है, पर यहाँ C11 का भागफल -4 नहीं है। C का शेषफल -2 है।
- C: सही। भाजक शून्य नहीं है और भागफल निरूपित हो सकता है, इसलिए दी गई पहचान लागू होती है और -17 मिलता है।
- D: गलत। -4 ऋणात्मक अनंत की ओर नीचे का मान है; C11 का पूर्णांक भाग शून्य की ओर जाता है।

प्रश्न 5

उत्तर: B

1. शुरू में $c = 7$, $tag = 0$ हैं।
2. पहली शर्त $c < 5$ गलत है, इसलिए $tag = 1$ छोड़ दिया जाता है।
3. फिर $c \% 2$ का मान 1 है, इसलिए $c \% 2 == 1$ सही है। $tag = 2$ चलता है।
4. पिछली शाखा चुनी जा चुकी है, इसलिए अंतिम `else` छोड़ दिया जाता है। आउटपुट 2 है।

विकल्पों की जाँच

- A: गलत। 1 का असाइनमेंट उस शाखा में है जिसकी शर्त गलत है।
- B: सही। विषम संख्या वाली शर्त की शाखा चुनी जाती है।
- C: गलत। सही `else if` शाखा के बाद अंतिम `else` नहीं चलता।
- D: गलत। चलने वाला $tag = 2$ असाइनमेंट शुरूआती 0 को बदल देता है।

प्रश्न 6

उत्तर: A, B

1. a का मान 0 है, इसलिए तार्किक जाँच में वह गलत है। b का मान 6 है।
2. $!a$ अर्थात् $!0$ का मान 1 है।
3. $a \ || \ (b > 4)$ में बायाँ operand गलत है, इसलिए दायाँ operand जाँचा जाता है। $6 > 4$ सही है और तार्किक OR का मान 1 है।
4. $a \ \&\& \ (b > 0)$ में बायाँ operand गलत है, इसलिए दायाँ operand छोड़ दिया जाता है और तार्किक AND का मान 0 है।
5. $!(b == 6)$ में समानता सही है और उसका मान 1 है; उसे नकारने पर 0 मिलता है। केवल A और B का मान 1 है।

विकल्पों की जाँच

- A: सही। तार्किक निषेध शून्य को 1 में बदलता है।
- B: सही। कम से कम एक operand सही है, इसलिए $||$ का मान 1 है, गैर-शून्य चर का मान 6 नहीं।
- C: गलत। बायाँ operand गलत होने पर $\&\&$ दाएँ operand को जाँचे बिना 0 देता है।
- D: गलत। समानता $b == 6$ सही है, इसलिए उसका निषेध 0 है।

प्रश्न 7

उत्तर: 4

1. आरंभीकरण से $\text{count} = 0$ और लूप का $k = 3$ होता है।
2. $k = 3$ पर शर्त सही है: $\text{count} 1$ होता है; अपडेट से $k = 6$ ।
3. $k = 6$ पर शर्त सही है: $\text{count} 2$ होता है; अपडेट से $k = 9$ ।
4. $k = 9$ पर शर्त सही है: $\text{count} 3$ होता है; अपडेट से $k = 12$ ।
5. $k = 12$ पर भी शर्त सही है क्योंकि अंतिम सीमा शामिल है: $\text{count} 4$ होता है; अपडेट से $k = 15$ ।
6. $k = 15$ पर शर्त गलत है। चार body और चार अपडेट हुए, पर शर्त पाँच बार जाँची गई। माँगी गई body की संख्या 4 है।

3 में शामिल अंतिम मान 12 छोड़ दिया गया है। 5 में अंतिम असफल शर्त को भी body चलने जैसा गिना गया है। 10 में 3-3 की दूरी वाले वास्तविक मानों के बजाय 3 से 12 तक हर पूर्णांक गिना गया है।

प्रश्न 8

उत्तर: D

1. शुरू में $n = 19$, $\text{sum} = 0$; $19 > 0$ सही है।
2. पहली body: $19 \% 3 = 1$, इसलिए $\text{sum} = 1$; फिर $n /= 3$ से $19 / 3 = 6$ रखा जाता है।
3. दूसरी जाँच $6 > 0$ सही है। $6 \% 3 = 0$, इसलिए $\text{sum} 1$ रहता है; पूर्णांक भाग से $n = 2$ ।
4. तीसरी जाँच $2 > 0$ सही है। $2 \% 3 = 2$, इसलिए $\text{sum} = 3$; पूर्णांक भाग से $n = 0$ ।
5. $0 > 0$ गलत है, इसलिए लूप समाप्त होता है। तीन body और चार जाँच हुई; छपने वाली मात्रा शेषफलों का योग 3 है।

विकल्पों की जाँच

- A: गलत। 4 शर्त की जाँचों की संख्या है, शेषफलों का योग नहीं। अंतिम असफल जाँच कुछ नहीं जोड़ती।
- B: गलत। 6 पहली बार अपडेट हुए n का मान है, sum से छपने वाला अंतिम मान नहीं।
- C: गलत। शुरुआती 19 को sum में नहीं जोड़ा जाता; हर body में केवल $n \% 3$ जुड़ता है।
- D: सही। जुड़े शेषफल 1, 0 और 2 हैं, जिनका योग 3 है।

प्रश्न 9

उत्तर: A, C, D

1. $\text{sum} = 0$ से शुरू करें, फिर $i = 0$ आरंभ होता है।
2. $i = 0$ पर शर्त $0 < 4$ सही है। $2 * 0 = 0$ जोड़ने पर $\text{sum} = 0$ रहता है; अपडेट से $i = 1$ ।
3. $i = 1$ पर शर्त $1 < 4$ सही है। 2 जोड़ने पर $\text{sum} = 2$; अपडेट से $i = 2$ ।
4. $i = 2$ पर शर्त $2 < 4$ सही है। 4 जोड़ने पर $\text{sum} = 6$; अपडेट से $i = 3$ ।
5. $i = 3$ पर शर्त $3 < 4$ सही है। 6 जोड़ने पर $\text{sum} = 12$; अपडेट से $i = 4$ ।
6. $i = 4$ पर शर्त $4 < 4$ गलत है। न पाँचवीं body चलती है, न पाँचवाँ अपडेट। body की संख्या 4, अपडेट की संख्या 4 और जाँच की संख्या 5 है। आउटपुट 12 4 है; सही कथन A, C और D हैं।

विकल्पों की जाँच

- A: सही। body $i = 0, 1, 2, 3$ पर चलती है।
- B: गलत। चार सफल जाँच और एक अंतिम गलत जाँच मिलकर पाँच होती हैं।
- C: सही। जमा योग $0 + 2 + 4 + 6 = 12$ है।
- D: सही। हर पूरी हुई body के बाद अपडेट चलता है; 3 से 4 करने वाला अपडेट भी इन चार में शामिल है।

प्रश्न 10

उत्तर: B

1. caller में $v = 4$ है। $\text{bump}(v)$ का आर्गुमेंट व्यंजक 4 देता है।
2. helper का अलग पैरामीटर $x = 4$ से शुरू होता है। $x += 3$ से helper का $x = 7$ होता है।
3. $\text{return } x$ helper समाप्त करके 7 लौटाता है। caller इसे result में रखता है।
4. caller के v को कोई नया मान असाइन नहीं हुआ; वह 4 रहता है। आउटपुट क्रम v, result है, इसलिए जोड़ी 4 7 है।

विकल्पों की जाँच

- A: गलत। 7 7 मानता है कि helper के पैरामीटर का अपडेट caller के v को बदल देता है।
- B: सही। आर्गुमेंट की कॉपी मिलती है और लौटा 7 result में रखा जाता है।
- C: गलत। 4 4 caller का मान सही रखता है, लेकिन return से पहले helper का जोड़ छोड़ देता है।
- D: गलत। इसमें असाइनमेंट के बिना caller का v बदल दिया गया है और पुराने मान को लौटे परिणाम के रूप में रखा गया है।

प्रश्न 11

उत्तर: 10

1. पहली कॉल: पैरामीटर $n = 3$; initializer के अनुसार स्थानीय s का शुरुआती मान 1 है।
2. $k = 1, 2, 3$ पर 2, 4, 6 जोड़ें। s के क्रमिक मान 3, 7, 13 हैं। $k = 4$ पर शर्त गलत है, इसलिए 13 लौटता है और $r = 13$ रखा जाता है।
3. दूसरी कॉल: पैरामीटर $n = 1$; इस कॉल का नया स्थानीय s फिर 1 से शुरू होता है।
4. $k = 1$ पर 2 जोड़ने से $s = 3$ । अपडेट से $k = 2$; शर्त गलत है। 3 लौटता है और $t = 3$ रखा जाता है।
5. caller में $r - t = 13 - 3 = 10$ छपता है। उत्तर पूर्णांक 10 है।

13 पहली कॉल के बाद रुककर छपने वाले घटाव को छोड़ देता है। -2 तब आएगा जब दूसरी बार s को गलत ढंग से 13 से आगे बढ़ाकर 15 निकाला जाए और उसे 13 से घटाया जाए। 12 में दूसरी कॉल के लौटे 3 के बजाय उसका शुरुआती 1 घटाया गया है। केवल संख्या लिखना यह नहीं दिखाता कि दोनों स्थानीय आरंभीकरण का ट्रेस किया गया है।

प्रश्न 12

उत्तर: A, C, D

1. दिए गए -3 से 3 तक हर पूर्णांक के लिए $n < 0$ जाँचें। ऋणात्मक input पर return $-n$ चलता है; शून्य और धनात्मक input पर return $n + 1$ । ऋणात्मक शाखा का return कॉल समाप्त करता है, इसलिए बाद का return भी साथ नहीं चलता।
2. choose(-3) ऋणात्मक शाखा लेकर 3 लौटाता है।
3. choose(0) में ऋणात्मक होने की शर्त गलत है और $0 + 1 = 1$ लौटता है।
4. choose(2) भी अंतिम return तक पहुँचता है और $2 + 1 = 3$ देता है।
5. caller में value = -2 होने पर helper के अलग पैरामीटर को -2 मिलता है और वह 2 लौटाता है। caller का value -2 रहता है; result 2 होता है। इसलिए दिखाया गया प्रोग्राम 3 1 3 -2 2 छापता है।
6. सही कथन ठीक A, C और D हैं। बताए गए inputs -3, -2, -1, 0, 1, 2, 3 पर क्रमशः outputs 3, 2, 1, 1, 2, 3, 4 हैं; पूरी अंकगणित दिए गए छोटे पूर्णांकों की सीमा में है।

विकल्पों की जाँच

- A: सही। -3 पर $n < 0$ सही है और $-n$ का मान 3 है।
- B: गलत। शून्य ऋणात्मक नहीं है; वह return $n + 1$ तक पहुँचकर 1 लौटाता है।
- C: सही। धनात्मक 2 गैर-ऋणात्मक रास्ते से 3 लौटाता है।
- D: सही। helper को आर्गुमेंट का पूर्णांक मान मिलता है; गणना लौटाने से caller के value में असाइनमेंट नहीं होता। बताई गई पूरी input सीमा में यह सही है।

किस बात का फिर अभ्यास करें

प्रश्न 1-प्रश्न 2: वर्तमान अवस्था और पूर्णांक अंकगणित। प्रश्न 3-प्रश्न 4: व्यंजक के मान और signed भाग। प्रश्न 5-प्रश्न 6: चुनी गई शाखाएँ और तार्किक जाँच। प्रश्न 7-प्रश्न 9: लूप की सटीक अवस्थाएँ और गिनती। प्रश्न 10-प्रश्न 12: पैरामीटर, स्थानीय आरंभीकरण और return। गलत प्रश्न में वह पहली अवस्था सुधारें जहाँ आपका ट्रेस अलग हुआ, फिर वहाँ से आगे दोबारा चलें।

स्रोत टिप्पणी

C11 के अर्थ-नियमों का संदर्भ [WG14 का N1570 समिति मसौदा](#) है। असाइनमेंट, अंकगणित, तार्किक ऑपरेटर, लूप और फ़ंक्शन के लिए §§6.5, 6.8.4-6.8.6 और 6.9.1 देखें।

उपमा और उसकी सीमाएँ

ट्रेस को ऐसा बही-खाता समझें जिसमें हर बदलाव की अलग पंक्ति है। असाइनमेंट एक प्रविष्टि बदलता है, शाखा चुनती है कि कौन-सी पंक्तियाँ चलेंगी और लूप शर्त गलत होने तक पंक्तियों का समूह दोहराता है। फ़ंक्शन कॉल अपने कॉपी किए पैरामीटर और स्थानीय चरों के लिए अलग कार्यपत्र खोलती है, फिर एक परिणाम लौटाती है। बही-खाते मिलाने से दिखता है कि दो सही लगने वाले उत्तरों में अंतर कहाँ शुरू हुआ।

त्वरित संदर्भ

- 12 प्रश्न, 12 अंक; निःशुल्क और बिना समय-सीमा; सही पर +1, गलत या खाली पर 0।
- MCQ: एक विकल्प। MSQ: ठीक पूरा सही समुच्चय, कोई आंशिक अंक नहीं। NAT: सटीक पूर्णांक।
- न ऋणात्मक अंक, न उत्तीर्ण सीमा। आगे किस बात का अभ्यास करना है, यह हल से पहचानें।
- हर प्रोग्राम उसके अपने शुरुआती मानों से शुरू करें; पिछले प्रश्न उसकी अवस्था नहीं बदलते।
- असाइनमेंट वर्तमान अवस्था उपयोग करता है। अलग बढ़ोतरी वाले कथन अगले कथन से पहले पूरे होते हैं।
- C11 का signed पूर्णांक भाग शून्य की ओर कटता है; शेषफल से भाज्य का पुनर्निर्माण होता है।
- && और || short-circuit करते हैं। तार्किक परिणाम 0 या 1 होते हैं।
- सफल body, शर्त की जाँच और अपडेट अलग-अलग गिनें।
- पूर्णांक पैरामीटर caller के चर से अलग है। दिखाया हर स्थानीय initializer अपनी कॉल में फिर चलता है।
- printf के आर्गुमेंट बताए आउटपुट क्रम में पढ़ें; इस क्रम को मूल्यांकन के सामान्य क्रम की गारंटी न समझें।

[विषय सूची पर लौटें](#)