## C (Programming Language)

##### Q1. Which Code sample will eventually cause the computer to run out of memory?

- [x]

```c

while(1)

{

char \*smallString = (char \*) malloc(10);

}

```

- [ ]

```c

long long number = 1;

while(1)

number \*= 2;

```

- [ ]

```c

while(1)

{

char hugeString[1000000L];

memset(hugeString, 0, 1000000L);

}

```

- [ ]

```c

while(1)

{

long \*bigArray = (long \*) malloc(sizeof(long) \* 1000);

memset(bigArray, 1000000, 1000);

free(bigArray);

}

```

#### Q2. What will this code print on the screen?

```c

int f1 (int a, int b)

{

if (a > b)

{

printf("A is greater than B\n");

return 1;

}

else

{

printf("B is greater than A");

return 0;

}

}

main()

{

if (f1(20,10) || f1(10,20))

printf("C is fun!\n");

}

```

- [x]

```

A is greater then B

C is fun!

```

- [ ]

```

A is greater then B

B is greater then A

C is fun!

```

- [ ]

```

A is greater then B

B is greater then A

```

- [ ] Nothing is printed on Screen

#### Q3. What is the name for calling a function inside the same function?

- [x] recursion

- [ ] subfunction

- [ ] inner call

- [ ] infinite loop

#### Q4. What does the declaration of variable c2 demonstrate?

```c

main(){

char c1 ='a';

char c2 = c1+10;

}

```

- [x] character arithmetic

- [ ] undefined assignment

- [ ] type conversion

- [ ] invalid declaration

#### Q5. A pointer to void named vptr, has been set to point to a floating point variable named g. What is the valid way to dereference vptr to assign its pointed value to a float variable named f later in this program?

```c

float g;

void \*vptr=&g;

```

- [ ] f = \_(float \_)vptr;

- [ ] f = (float \\*)vptr;

- [x] f = \\*(float \\*)vptr;

- [ ] f = \\*(float)vptr;

#### Q6. What is this declaration an example of?

```c

struct s {

int i;

struct s \*s1;

struct s \*s2;

};

```

- [x] a node

- [ ] a linked list

- [ ] a stack

- [ ] a binary tree

#### Q7. A C header file is a file with extension .h that contains function declarations and macro definitons to be shared between several source files. Header files are listed using the preprocessing directive #include, and can have one of the following formats: #include &lt;fileA&gt; or #include "fileB". What is the difference between these two formats?

- [ ] The preprocessor will try to locate the fileA in same directory as the source file, and the fileB in a predetermined directory path.

- [ ] The preprocessor will try to locate the fileA in the fixed system directory. It will try to locate fileB in the directory path designated by the -l option added to the command line while compiling the source code.

- [ ] The file using fileA syntax must be system files, of unlimited number. fileB must be a user file at a maximun of one per source file.

- [x] The preprocessor will try to locate the fileA in a predetermined directory path. It will try to locate fileB in the same directory as the source file along with a custom directory path.

#### Q8. Using a for loop, how could you write a C code to count down from 10 to 1 and display each number on its own line?

- [ ]

```c

for (int i = 0; i>=0, i--){

printf("%d\n", i);

}//end of loop

```

- [ ]

```c

int i;

for (i=1; i<=10; i++){

printf("%d", i);

}

```

- [ ]

```c

int i = 10;

while (i>0){

printf("%d\n", i);

i--;

}

```

- [x]

```c

int i;

for (i= 10; i>0; i--){

printf("%d\n", i);

}// end of loop

```

#### Q9. What is not one of the reserved words in standard C?

- [ ] volatile

- [x] typeof

- [ ] register

- [ ] typedef

[Reference](https://www.ibm.com/docs/en/adfz/developer-for-zos/14.2.0?topic=programs-c-reserved-keywords)

#### Q10. What does the program shown below return?

```c

int main(){

int a=1, b=2, c=3, d=4;

int x = a;

if (a>b)

if (b<c) x=b;

else x=c;

return(x);

}

```

- [x] 1

- [ ] 3

- [ ] 2

- [ ] 0

#### Q11. Using the Union declaration below, how many bytes of memory space will the data of this type occupy?

```c

union Cars {

char make[20];

char model[30];

short year;

} car;

```

- [ ] 32

- [ ] 54

- [x] 30

- [ ] 52

#### Q12. In this code sample, what is not a problem for C compiler?

```c

main(){

constant int PI = 3.14;

printf("%f\n", pi);

}

```

- [x] The value of PI needs to be set to 3.141593, not 3.14

- [ ] The declaration of PI needs to say const, not constant.

- [ ] The data type of PI needs to be float not int.

- [ ] The printf statement needs to use PI, not pi.

#### Q13. Which is the smallest program to compile and run without errors?

- [ ] main()

- [ ] int main() {return 0;}

- [x] main() { }

- [ ] main() { ; }

[Reference](https://www.beningo.com/150-the-wolrds-shortest-c-program/)

#### Q14. What is optional in a function declaration?

- [ ] data type of parameters

- [ ] return type of function

- [x] parameter names

- [ ] number of parameters

#### Q15. C treats all devices, such as the display and the keyboard, as files. Which files opens automatically when a program executes?

- [x] stdout

- [ ] stdio.h

- [ ] default.h

- [ ] string.h

#### Q16. In which segment does dynamic memory allocation takes place?

- [ ] BSS Segment

- [ ] stack

- [x] heap

- [ ] data segment

[Reference](http://www.it.uc3m.es/pbasanta/asng/course\_notes/dynamic\_memory\_heap\_en.html#:~:text=The%20dynamic%20memory%20that%20is,reads%20a%20set%20of%20words.)

#### Q17. Which of the following do you use to deallocate memory?

- [ ] dalloc()

- [ ] dealloc()

- [ ] release()

- [x] free()

[Reference](https://www.tutorialspoint.com/c\_standard\_library/c\_function\_free.htm)

#### Q18. In C language what are the basic building blocks that are constructed together to write a program?

- [ ] keywords

- [ ] identifiers

- [x] tokens

- [ ] functions

#### Q19. When is memory for a variable allocated?

- [ ] during the assigment of the variable

- [ ] during the initialization of the variable

- [x] during the declaration of the variable

- [ ] during the definition of the variable

#### Q20. By default c uses the call by value method to pass arguments to functions. How can you invoke the call by reference method?

- [x] by using pointers

- [ ] by declaring functions separately from defining them

- [ ] by using recursive functions

- [ ] by using global variables

#### Q21. A union allows you to store different `\_\_\_` in the same `\_\_\_`.

- [ ] Objects; Structure

- [ ] Variables; Declaration

- [x] Data types; Memory location

- [ ] Arrays; Header file

#### Q22. What is the output of this program?

```c

main() {

char c1='a' , c2='A';

int i=c2-c1;

printf("%d", i);

}

```

- [ ] 32

- [ ] Runtime error

- [x] -32

- [ ] 0

#### Q23. What is the difference between scanf() and sscanf() functions?

- [ ] The scanf() function reads data formatted as a string; The sscanf() function reads string input from the screen.

- [x] The scanf() function reads formatted data from the keyboard; The sscanf() function reads formatted input from a string.

- [ ] The scanf() function reads string data from the keyboard; The sscanf() function reads string data from a string.

- [ ] The scanf() function reads formatted data from a file; The sscanf() function reads input from a selected string

#### Q24. What is not a valid command with this declaration?

```c

char \*string[20] = { "one", "two", "three"};

```

- [ ] `printf("%c", string[1][2]);`

- [x] `printf("%s", string[1][2]);`

- [ ] `printf("%s", string[1]);`

- [ ] `printf(string[1]);`

#### Q25. What is the expression player->name equivalent to?

- [ ] `player.name`

- [x] `(\\*player).name`

- [ ] `\\*player.name`

- [ ] `player.\\*name`

#### Q26. Which program will compile and run without errors?

- [ ]

```c

main() {

for(i=0; i<10; i++) ;

}

```

- [x]

```c

main() {

int i=0;

for(; i<10; i++) ;

}

```

- [ ]

```c

main() {

int i;

for(i=0; i<j; i++) ;

}

```

- [ ]

```c

main() {

int i;

for (i= 10; i<10; i++)

}

```

#### Q27. What does this function call return?

```c

1 main() { float x = f1(10, 5); }

2 float f1(int a, int b) { return (a/b); }

```

- [ ] 2

- [ ] 2.000000

- [ ] a runtime error

- [x] a compiler error

#### Q28. What does this program create?

```c

#include <stdio.h>

int main() {

int \*p = NULL;

return 0;

}

```

- [ ] a runtime error

- [x] a NULL pointer

- [ ] a compile error

- [ ] a void pointer

#### Q29. What is an alternative way to write the expression (\\*x).y?

- [ ] There is no equivalent.

- [x] x->y

- [ ] \\*x->y

- [ ] y->x

#### Q30. Compile time errors are static errors that can be found where in the code?

- [x] in declarations and definitions

- [ ] in functions and expressions

- [ ] in syntax and semantics

- [ ] in objects and statements

#### Q31. File input and output (I/O) in C is heavily based on the way it is done `\_\_\_`?

- [x] in Unix

- [ ] in C++

- [ ] in C#

- [ ] in DOS

#### Q32. What does the strcmp(str1, str2); function return?

- [x] 0 if str1 and str2 are the same, a negative number if str1 is less than str2, a positive number if str1 is greater than str2

- [ ] true (1) if str1 and str2 are the same, false (0) if str1 and str2 are not the same

- [ ] true (1) if str1 and str2 are the same, NULL if str1 and str2 are not the same

- [ ] 0 if str1 and str2 are the same, a negative number if str2 is less than str1, a positive number if str2 is greater than str1

#### Q33. What is the output of this program?

```c

int a=10, b=20;

int f1(a) { return(a\*b); }

main() {

printf("%d", f1(5));

}

```

- [x] 100

- [ ] 200

- [ ] 5

- [ ] 50

#### Q34. Which is \_not\_ a correct way to declare a string variable?

- [ ] `char \*string = "Hello World";`

- [x] `char string = "Hello World";`

- [ ] `char string[20] = {'H', 'e', 'l', 'l', 'o', ' ', 'W', 'o', 'r', 'l', 'd'};`

- [ ] `char string[] = "Hello World";`

#### Q35. Which choice is an include guard for the header file mylib.h?

- [ ]

```c

#ifdef MYLIB\_H

#undef MYLIB\_H

// mylib.h content

#endif /\* MYLIB\_H \*/

```

- [x]

```c

#ifndef MYLIB\_H

#define MYLIB\_H

// mylib.h content

#endif /\* MYLIB\_H \*/

```

- [ ]

```c

#define MYLIB\_H

#include "mylib.h"

#undef MYLIB\_H

```

- [ ]

```c

#ifdef MYLIB\_H

#define MYLIB\_H

// mylib.h content

#endif /\* MYLIB\_H \*/

```

#### Q36. How many times does the code inside the while loop get executed in this program?

```c

main(){

int x=1;

while(x++<100){

x\*=x;

if(x<10) continue;

if(x>50) break

}

}

```

- [ ] 100

- [x] 3

- [ ] 5

- [ ] 50

#### Q37. File input and output (I/O) in C is done through what?

- [ ] syntax-driven components

- [ ] native interfaces

- [ ] system objects

- [x] function calls

#### Q38. Directives are translated by the?

- [x] Pre-processor

- [ ] Compiler

- [ ] Linker

- [ ] Editor

#### Q39. The main loop structures in C programming are the for loop, the while loop, and which other loop?

- [x] do...while

- [ ] for...in

- [ ] repeat...until

- [ ] do...until

#### Q40. By default, C Functions are what type of functions?

- [ ] global

- [ ] static

- [x] library

- [ ] system

#### Q41. You have written a function that you want to include as a member of structure a. How is such as structure member defiened?

- [x]

```c

struct a {

void \*f1;

};

```

- [ ]

```c

struct a {

void (\*f1)();

};

```

- [ ]

```c

struct a {

\*(void \*f1)();

};

```

- [ ]

```c

struct a {

void \*f1();

};

```

#### Q42. A Stack data structure allows all data operations at one end only, making it what kind of an implementation?

- [ ] FIFO

- [x] LIFO

- [ ] LILO

- [ ] LOLI

#### Q43. What does this program display?

```c

main(){

char \*p = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";

int i;

for (i=0;i<5;i++) \*p++; \*p++;

printf("%c",\*p++);

}

```

- [ ] K

- [ ] M

- [ ] H

- [x] G

#### Q44. Describe the relationship between lvalue and rvalue.

- [ ] An lvalue may appear only on the left-hand side of an assignment; an rvalue may appear only on the right-hand side.

- [ ] An lvalue may appear only on the left-hand side of an assignment; an rvalue may appear on either the left-hand or right-hand side.

- [ ] An lvaue and an rvalue may appear on either left-hand or right-hand side of an assignment.

- [x] An lvalue may appear on the left-hand or right-hand side of an assignment; an rvalue may appear only on the right-hand side.

#### Q45. Which operator is used to access the address of a variable?

- [ ] `%`

- [ ] `\*\*`

- [ ] `\*`

- [x] `&`

#### Q46. Which add function properly returns the updated value of result?

- [x]

```c

void add (int a, int b, int \*result)

{

\*result = a+b;

}

main()

{

int a = 10;

int b = 20;

int result = 0;

add(a,b,&result);

}

```

- [ ]

```c

void add (int a, int b, int result)

{

result = a+b;

}

main()

{

int a = 10;

int b = 20;

int result = 0;

add(a,b,result);

}

```

- [ ]

```c

void add (int a, int b, int \*result)

{

result = a+b;

}

main()

{

int a = 10;

int b = 20;

int result = 0;

add(a,b,result);

}

```

- [ ]

```c

void add (int \*a, int \*b, int \*result)

{

result = a+b;

}

main()

{

int a = 10;

int b = 20;

int result = 0;

add(\*a,\*b,\*result);

}

```

#### Q47. Consider the number of the Fibonacci series below 100: 0,1,1,2,3,5,8,13,21,34,55,89. Which piece of code outputs the sequence?

- [ ]

```c

void fibonacci(int a, int b)

{

int c = a+b;

if(a>100)

return;

printf("%d", a);

fibonacci(a,b);

}

int main()

{

fibonacci(0,1);

}

```

- [ ]

```c

void fibonacci(int a, int b)

{

int c = a+b;

if(a>100)

return;

printf("%d", b);

fibonacci(a,c);

}

int main()

{

fibonacci(0,1);

}

```

- [x]

```c

void fibonacci(int a, int b)

{

int c = a+b;

if(a>100)

return;

printf("%d", a);

fibonacci(b,c);

}

int main()

{

fibonacci(0,1);

}

```

- [ ]

```c

void fibonacci(int a, int b)

{

int c = a+b;

if(a>100)

return;

printf("%d", c);

fibonacci(b,c);

}

int main()

{

fibonacci(0,1);

}

```

#### Q48. Which is \_not\_ a storage class specifier?

- [x] `intern`

- [ ] `extern`

- [ ] `register`

- [ ] `static`

[Reference](https://en.cppreference.com/w/cpp/language/storage\_duration)

#### Q49. Which line of code, after execution, results in `i` having the value of 1?

- [ ] `for(i=1; i<=1; i++);`

- [ ] `for(i=1; i=10; i++);`

- [x] `for(i=1; i==10; i++);`

- [ ] `for(i=10; i>=1; i--);`

#### Q50. What is the value of variable c at the end of this program?

```

1 main() {

2 int a, b, c;

3 a=10; b=50;

4 c=a \* b % a;

5 }

```

- [ ] 50

- [ ] 5

- [x] 0

- [ ] 500

### Q51. What is \_not\_ one of the basic data types in C

- [ ] long double

- [ ] unsigned char

- [x] array

- [ ] float

### Q52. What is the member access operator for a structure?

- [ ] ,

- [ ] []

- [x] .

- [ ] :

#### Q53. What standard data type provides the smallest storage size and can be used in computations?

- [x] char

- [ ] float

- [ ] int

- [ ] short

#### Q54. what does the ctype tolower() function do?

- [ ] It returns TRUE for lowercase letters of the alphabet.

- [ ] It ensures that text output uses only ASCII values (0 through 127).

- [ ] It returns FALSE for lowercase letters of the alphabet.

- [x] It converts an uppercase letter of the alphabet to lowercase.

#### Q55. Void pointer \_vptr\_ is assigned the address of float variable \_g\_. What is a valid way to dereference \_vptr\_ to assign its pointed value to a float variable named \_f\_ later in the program?

```c

float g;

void \*vptr=&g;

```

- [ ] f=(float \\*)vptr;

- [x] f=\_(float \_)vptr;

- [ ] f=\\*(float)vptr;

- [ ] f=(float)\\*vptr;

#### Q56. Using a \_for\_ loop, how would you write C code to count down from 10 to 1 and display each number on its own line?

- [ ]

```c

for(int i=10;1>0;) {

printf("%d\n", i);

i--;

}

```

- [x]

```c

for (int i=10; i>0; i--)

printf("%d\n", i);

```

- [ ]

```c

for (int i=1; i<=10; i++)

printf("%d", i);

```

- [ ]

```c

for (int i=10; i>=0; i--)

printf("%d\n", i);

```