1) What is a function pointer in C?

**a) A pointer that points to a function**

b) A pointer that points to a variable

c) A pointer that points to an array

d) A pointer that points to a structure

2)How do you declare a function pointer?

**a) int (\*ptr)()**

b) int \*\*ptr()

c) int (&ptr)()

d) int ptr()

3)Which keyword is used to define an inline function in C?

**a) inline**

b) function

c) inlinefunc

d) def

4)What is the purpose of using function pointers in C?

**a) To store the address of a function**

b) To store the value of a function

c) To store the result of a function

d) To store the input of a function

5)What is the syntax to call a function using a function pointer?

a) functionPointer()

**b) (\*functionPointer)()**

c) \*functionPointer()

d) &functionPointer()

6)Which of the following is true about function pointers?

a) Function pointers can be used to create recursive functions.

b) Function pointers cannot be passed as arguments to other functions.

c) Function pointers can only point to functions within the same file.

**d) Function pointers can be used to create callback mechanisms.**

7)How do you declare a function that takes a function pointer as an argument?

a) void myFunction(int \*func)

b) void myFunction(int func())

**c) void myFunction(int (\*func)())**

d) void myFunction(int func)

8)What is the output of the following code?

#include <stdio.h>

int add(int a, int b)

{

return a + b;

}

int main()

{

int (\*ptr)(int, int) = add;

printf("%d\n", ptr(3, 4));

return 0;

}

a) 3

b) 4

**c) 7**

d) 12

9) Which keyword is used to define a function as an inline function?

**a) inline**

b) function

c) inlinefunc

d) def

10) When should you use an inline function?

a) When the function has a large number of statements

b) When the function is recursive

**c) When the function is called frequently and has a small body**

d) When the function is defined in a separate source file

11) What is the purpose of using the "inline" keyword in C?

**a) To optimize the execution time of the function**

b) To reduce the memory usage of the function

c) To indicate that the function should be called only once

d) To indicate that the function should be called multiple times

12) What happens if a function declared as "inline" is not inlined by the compiler?

a) The program will fail to compile

b) The function will be automatically optimized by the compiler

**c) The function will be called as a regular function**

d) The program will crash at runtime

13) What is the output of the following code?

#include <stdio.h>

inline int square(int x)

{

return x \* x;

}

int main()

{

printf("%d\n", square(5));

return 0;

}

a) 5

b) 10

**c) 25**

d) 50

14) How do you define an inline function?

**a) By using the "inline" keyword before the function name**

b) By using the "def" keyword before the function name

c) By using the "function" keyword before the function name

d) By using the "inlinefunc" keyword before the function name

15) Which of the following is true about inline functions?

a) Inline functions cannot have any arguments

b) Inline functions cannot have a return type

**c) Inline functions are expanded at compile time**

d) Inline functions are always faster than regular functions

16) What is the output of the following code?

#include <stdio.h>

inline int max(int a, int b)

{

return (a > b) ? a : b;

}

int main()

{

int x = 10;

int y = 20;

printf("%d\n", max(x, y));

return 0;

}

a) 10

**b) 20**

c) 30

d) Compiler error

17)Can an inline function have recursion?

a) Yes

**b) No**

18)Which of the following is true about function pointers?

a) Function pointers can only point to functions with a void return type.

b) Function pointers cannot be used to call functions with arguments.

**c) Function pointers can be used to create generic functions.**

d) Function pointers are used to define new data types.

19)What is the output of the following code?

#include <stdio.h>

void printHello()

{

printf("Hello\n");

}

int main()

{

void (\*ptr)() = printHello;

ptr();

return 0;

}

**a) Hello**

b) World

c) Error

d) No output

20) What is the syntax to define a function pointer type?

a) typedef int \*funcPtr;

b) typedef int funcPtr();

**c) typedef int (\*funcPtr)();**

d) typedef int &funcPtr();

21) How do you assign a function to a function pointer?

**a) ptr = functionName;**

b) \*ptr = functionName;

c) &ptr = functionName;

d) \*\*ptr = functionName;

22) What is the output of the following code?

#include <stdio.h>

int add(int a, int b)

{

return a + b;

}

int subtract(int a, int b)

{

return a - b;

}

int main()

{

int (\*ptr)(int, int);

ptr = add;

printf("%d\n", ptr(3, 4));

ptr = subtract;

printf("%d\n", ptr(3, 4));

return 0;

}

**a) 7, 1**

b) 7, -1

c) 1, 7

d) -1, 7

23) Which of the following is true about inline functions?

a) Inline functions cannot be called from other functions.

b) Inline functions always have a return statement.

c) Inline functions are expanded at runtime.

**d) Inline functions may not always be inlined by the compiler.**

24) What is the output of the following code?

#include <stdio.h>

inline int multiply(int a, int b)

{

return a \* b;

}

int main()

{

int x = 5;

int y = 2;

printf("%d\n", multiply(x, y));

return 0;

}

a) 5

b) 2

**c) 10**

d) 7

25) Can you pass a function pointer as an argument to a function?

**a) Yes**

b) No

26) What is the output of the following code?

#include <stdio.h>

void square(int \*x)

{

\*x = (\*x) \* (\*x);

}

int main()

{

int num = 5;

void (\*ptr)(int \*) = square;

ptr(&num);

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

return 0;

}

a) 5

b) 10

**c) 25**

d) 50

27) What is the purpose of using function pointers in C?

a) To reduce code size

b) To improve code readability

**c) To enable dynamic function dispatch**

d) To simplify the syntax of function calls

28) What is the output of the following code?

#include <stdio.h>

int add(int a, int b)

{

return a + b;

}

int subtract(int a, int b)

{

return a - b;

}

int main()

{

int (\*ptr[2])(int, int) = {add, subtract};

printf("%d\n", ptr[0](3, 4));

printf("%d\n", ptr[1](3, 4));

return 0;

}

**a) 7, 1**

b) 7, -1

c) 1, 7

d) -1, 7

29) Which of the following is true about function pointers?

a) Function pointers cannot be used as arguments to other functions.

b) Function pointers cannot be stored in arrays.

**c) Function pointers can be used to implement callbacks.**

d) Function pointers can only point to functions with a void return type.

30) What is the output of the following code?

#include <stdio.h>

void printMessage()

{

printf("Hello, World!\n");

}

int main()

{

void (\*ptr)();

ptr = printMessage;

ptr();

return 0;

}

**a) Hello, World!**

b) World, Hello!

c) Error

d) No output

31)How do you pass a function pointer as an argument to a function?

a) By using the function name

**b) By using the function address**

c) By using the function return value

d) By using the function type

32)What is the output of the following code?

#include <stdio.h>

inline int cube(int x)

{

return x \* x \* x;

}

int main()

{

printf("%d\n", cube(3));

return 0;

}

a) 3

b) 9

**c) 27**

d) 81

33)Can an inline function have a variable number of arguments?

a) Yes

**b) No**

34)What is the output of the following code?

#include <stdio.h>

int multiply(int a, int b)

{

return a \* b;

}

int main()

{

int (\*ptr)(int, int) = multiply;

printf("%d\n", (\*ptr)(3, 4));

return 0;

}

a) 3

b) 4

**c) 7**

d) 12

35) Which of the following is true about inline functions?

a) Inline functions are always faster than regular functions.

**b) Inline functions cannot be recursive.**

c) Inline functions must be defined in a separate source file.

d) Inline functions can have a variable number of arguments.