C MASTERY

STUDYBOOSTERS
ELEVATE YOUR EXAM GAME

Short Notes

UNIT 1 Basics and introduction to C

- The C character set: C uses ASCII character set to represent characters. This includes letters, digits, special characters, and control characters.
- Identifiers and keywords: Identifiers are names given to variables, functions, and other user-defined entities in C. Keywords, on the other hand, are reserved words that have predefined meanings in the language.
- Data types: C supports several data types, including integer, floating-point, character, and void. These data types can be combined to form complex data structures.
- Constants and variables: Constants are values that remain fixed throughout the program, while variables are values that can be modified during program execution.
- Expressions: An expression is a combination of variables, constants, and operators that result in a value. Expressions can be used to perform calculations, comparisons, and other operations.
- Arithmetic operators: C supports several arithmetic operators, including addition (+), subtraction (-), multiplication (*), division (/), and modulus (%).
- Unary operators: Unary operators are operators that operate on a single operand. Examples include the increment (++), decrement (--), and negation (-) operators.
- Relational operators: Relational operators are used to compare two values. Examples include the less than (<), greater than (>), less than or equal to (<=), and greater than or equal to (>=) operators.
- Logical operators: Logical operators are used to combine multiple conditions in a single expression. Examples include the AND (&&), OR (||), and NOT (!) operators.
- Assignment operators: Assignment operators are used to assign a value to a variable. Examples include the equal to (=) operator and the compound assignment operators (+=, -=, *=, /=, %=).
- Conditional operator: The conditional operator (?:) is a shorthand way of writing an if-else statement. It evaluates a condition and returns one of two values, depending on whether the condition is true or false.
- Bitwise operators: Bitwise operators are used to perform operations on the individual bits of a number. Examples include the AND (&), OR (|), XOR (^), left shift (<<), and right shift (>>) operators.

UNIT 2 Control Structure and Input/Output functions

- If statement: The if statement is used to execute a block of code if a condition is true. It can be followed by an optional else statement to execute a different block of code if the condition is false.
- If else statement: The if-else statement is used to execute one block of code if a condition is true, and another block of code if the condition is false.
- Switch case statement: The switch statement is used to execute different blocks of code based on the value of a variable or expression. Each case statement specifies a different value to compare against.
- While loop: The while loop is used to repeatedly execute a block of code as long as a condition is true.
- For loop: The for loop is used to execute a block of code a specified number of times. It consists of an initialization statement, a condition statement, and an increment statement.
- Do-while loop: The do-while loop is similar to the while loop, but it executes the block of code at least once, regardless of the condition.
- Break and continue statements: The break statement is used to exit a loop or switch statement. The continue statement is used to skip the current iteration of a loop and continue with the next iteration.
- Goto statement: The goto statement is used to jump to a labeled statement in the code. It is generally discouraged, as it can make code harder to read and debug.
- Return statement: The return statement is used to exit a function and return a value to the calling function.
- Type conversion and type modifiers: Type conversion is the process of converting a value from one data type to another. C also supports type modifiers, such as const, volatile, and static, which affect the behavior of variables.
- Designing structured programs in C: Structured programming is a programming paradigm that emphasizes the use of structured control structures, such as if statements and loops, to improve the clarity and maintainability of code.
- Formatted and unformatted Input/Output functions: C provides several functions for input and output, including printf() for formatted output, scanf() for formatted input, puts() for unformatted output, and gets() for unformatted input.

UNIT 3 User Defined functions and Storage classes

- Function prototypes: A function prototype is a declaration of a function that specifies the function's name, return type, and parameter list. It is used to tell the compiler about the function before it is called.
- Function definition: A function definition is the actual implementation of the function. It consists of a function header, which includes the function's name and parameter list, and a function body, which contains the statements that are executed when the function is called.
- Function call: A function call is the process of executing a function. It involves passing arguments to the function, which can be done by value or by reference.
- Passing arguments by value: When arguments are passed by value, a copy of the argument's value is made and passed to the function. Any changes made to the argument within the function do not affect the original value.
- Passing arguments by reference: When arguments are passed by reference, a reference to the argument's memory address is passed to the function. Any changes made to the argument within the function affect the original value.
- Math library functions: C provides a math library that contains several functions for performing mathematical calculations, such as trigonometric functions, logarithmic functions, and exponential functions.
- Recursive functions: A recursive function is a function that calls itself. It can be used to solve problems that can be broken down into smaller subproblems.
- Scope rules: Scope refers to the visibility of variables within a program. Local variables are visible only within the function in which they are defined, while global variables are visible throughout the entire program.
- Storage classes: C provides four storage classes for variables: auto, extern, register, and static.
- Auto storage class: Variables declared with the auto storage class are allocated on the stack and are destroyed when the function in which they are defined returns.
- Extern storage class: Variables declared with the extern storage class are declared in one file and can be used in other files.
- Register storage class: Variables declared with the register storage class are stored in registers instead of memory, which can improve performance.
- Static storage class: Variables declared with the static storage class retain their values between function calls and are only initialized once.

UNIT 4 Pointers & Arrays in C

Pointers:

- Pointer declaration and initialization: A pointer is a variable that stores the memory address of another variable. Pointers are declared using the * operator, and initialized using the & operator to get the address of a variable.
- Pointer operators: The * operator is used to dereference a pointer and access the value stored at the memory address it points to. The & operator is used to get the memory address of a variable.
- Pointer expressions and arithmetic: Pointer arithmetic involves adding or subtracting integers to a pointer to move it to a different memory address. Pointer expressions can be used to access arrays and structures.
- Operations on pointers: Pointers can be compared for equality or inequality, and can be assigned to or from other pointers of the same type.
- Passing pointers to a function: Pointers can be passed as arguments to functions, which allows the function to access and modify variables in the calling function.
- Pointer and one dimensional array: One dimensional arrays in C are implemented as pointers to the first element of the array.
- Pointer to a group of one dimensional arrays: An array of pointers can be used to represent a group of one dimensional arrays.
- Array of pointers: An array of pointers can be used to represent a two dimensional array.
- Types of pointers: Dangling pointers are pointers that point to memory that has been deallocated. Wild pointers are uninitialized pointers. Null pointers are pointers that point to the null memory address. Generic (void) pointers can point to any type of data.

Arrays:

- Declaring and initializing arrays in C: Arrays in C are declared using square brackets, and can be initialized with a list of values enclosed in curly braces.
- Defining and processing 1D and 2D arrays: One dimensional arrays store a sequence of values, while two dimensional arrays store values in a grid or table. Arrays can be processed using loops and array indexing.
- Array applications: Arrays can be used to store and process large amounts of data, such as sensor readings or image data.
- Passing arrays to functions: Arrays can be passed as arguments to functions, allowing the function to process the array data.

- Inserting and deleting elements of an array: Elements can be inserted or deleted from an array by shifting the remaining elements and changing the size of the array.
- Searching including linear and binary search methods: Linear search involves checking each element of an array in order, while binary search involves dividing the array in half and checking the middle element.
- Sorting of array using bubble sort: Bubble sort is a simple sorting algorithm that repeatedly swaps adjacent elements if they are in the wrong order.

UNIT 5 Dynamic Memory Management & File I/O

Dynamic Memory Management:

- Dynamic Memory Management functions: C provides several functions for dynamic memory allocation, including malloc(), calloc(), realloc(), and free(). These functions allow programs to allocate and deallocate memory at runtime.
- Malloc(): malloc() is used to allocate a block of memory of a specified size. The function returns a pointer to the first byte of the allocated memory.
- Calloc(): calloc() is used to allocate a block of memory for an array of elements, each of which is initialized to zero.
- Realloc(): realloc() is used to resize a previously allocated block of memory.
- Free(): free() is used to deallocate a block of memory that was previously allocated with malloc(), calloc(), or realloc().
- Memory leak: A memory leak occurs when a program fails to deallocate memory that it has allocated. This can result in the program using more and more memory until the system runs out of memory.

File I/O:

- The FILE structure: C provides the FILE structure for working with files. The FILE structure contains information about the file, such as its name, mode, and location in the file.
- Opening and closing files: C provides several functions for opening and closing files, including fopen(), fclose(), and feof(). fopen() is used to open a file, fclose() is used to close a file, and feof() is used to check whether the end of the file has been reached.

- Text and binary files: Text files contain ASCII or Unicode characters, while binary files contain binary data.
- Reading, writing and appending files: C provides several functions for reading from and writing to files, including fprintf(), fscanf(), fgets(), and fputs(). These functions allow programs to read and write text or binary data to and from files.
- Random access of files: C provides several functions for randomly accessing files, including fseek() and ftell(). These functions allow programs to move to a specific location in a file and read or write data from that location.

UNIT 6 Strings,Derived types including structures and unions

Strings:

- A string is a sequence of characters stored in an array with a null character ('\0') at the end.
- Defining and initializing strings: Strings can be defined and initialized using the char data type.
- Reading and writing a string: C provides several functions for reading and writing strings, including gets(), puts(), scanf(), and printf().
- Processing of strings: C provides several functions for processing strings, including strlen(), strcat(), strcpy(), and strcmp(). These functions allow programs to manipulate strings.
- Character arithmetic: C allows arithmetic operations to be performed on characters. Characters are represented as integers using ASCII codes.

Derived types:

- Structures and unions are derived types in C.
- Declaration of a structure: A structure is declared using the struct keyword.
- Definition and initialization of structures: A structure can be defined and initialized using curly braces {}.
- Accessing structures: The members of a structure can be accessed using the dot operator (.).

- Structures and pointers: Pointers can be used to point to structures. The arrow operator (->) is used to access the members of a structure through a pointer.
- Nested structures: A structure can contain another structure as a member.
- Declaration of a union: A union is declared using the union keyword. A union is similar to a structure, but all of its members share the same memory location.

UNIT 1 Basics and introduction to C

Easy

i. Which of the following is NOT a valid C data type?

b) int
c) float
d) boolean
e) double

i. What is the difference between a constant and a variable in C?

a) A constant is a fixed value that cannot be changed, whereas a variable is a value that can be assigned and modified.
b) A constant is a value that can be assigned and modified, whereas a variable is a fixed value that cannot be changed.
c) A constant and a variable are both the same thing in C.
d) None of the above.

ii. Which of the following is a valid C identifier?

a) 2variable
b) variable2
c) var-iable
d) variable$

iii. Which of the following is a C keyword?

a) main
b) function
c) variable
d) loop

iv. Which of the following is NOT a valid C operator?

a) ++
b) --
c) **
d) /

v. What is the result of the expression 5 % 2?

a) 2.5
b) 2
c) 2.0
d) 2.5

vi. Which of the following is a logical operator in C?

a) &
b) |
c) !
d) +

vii. What is the value of x after the following code is executed: x = 10; x += 5;

a) 5
b) 10
c) 15
d) 20

viii. Which of the following is a bitwise operator in C?

a) &&
b) ||
c) ^
d) >

ix. What is the value of y after the following code is executed: x = 5; y = (x > 3) ? 10 : 20;

a) 5
b) 10
c) 15
d) 20

Medium

i. What is the difference between a float and a double in C?

b) A float is a data type that can store decimal numbers with less precision than a double.
c) A double is a data type that can store decimal numbers with less precision than a float.

d) Both float and double are the same data type in C.
e) None of the above.

i. Which of the following is a valid C constant?
a) 3.14f
b) 'A'
c) "hello"
d) all of the above

ii. Which of the following is a valid C expression?
a) 2 + 3 * 4
b) (2 + 3) * 4
c) 2 + (3 * 4)
d) all of the above

iii. What is the result of the expression (2 == 2) && (3 != 4)?
a) true
b) false
c) undefined
d) none of the above

iv. Which of the following is a valid C conditional operator?
a) &&
b) ||
c) !
d) ?:

v. What is the value of x after the following code is executed: x = 10; x /= 3;
a) 3.33
b) 3
c) 3.0
d) 3.33...

vi. Which of the following is NOT a valid C bitwise operator?
a) &
b) |
c) ~
d) !

vii. What is the value of x after the following code is executed: x = 5; x *= 2 + 3;
a) 25
b) 10
c) 15
d) 13

viii. Which of the following is a valid C unary operator?

a) ++
b) --
c) !
d) all of the above

ix. Which of the following is a valid C keyword for defining a function?

a) main
b) function
c) void
d) int

Hard

i. Which of the following is NOT a valid C data type modifier?

b) signed
b) unsigned
c) const
d) float

i. What is the value of x after the following code is executed: x = 5; x = x++ + ++x;

a) 10
b) 11
c) 12
d) undefined

ii. Which of the following is a valid C bit shift operator?

a) <<
b) >>
c) <>
d) ><

iii. What is the difference between a static variable and an automatic variable in C?

a) A static variable is a variable that retains its value between function calls, whereas an automatic variable is a variable that is destroyed when the function ends.
b) A static variable is a variable that is declared with the static keyword, whereas an automatic variable is a variable that is declared without any keyword.
c) Both static and automatic variables are the same thing in C.
d) None of the above.

iv. Which of the following is a valid C function prototype?

a) void function();

b) function();
c) function(void);
d) all of the above

v. What is the value of x after the following code is executed: x = 5; x = x-- - --x;

a) 0
b) 1
c) 2
d) undefined

vi. Which of the following is a valid C preprocessor directive?

a) #define
b) #include
c) #ifdef
d) all of the above

vii. What is the value of x after the following code is executed: x = 10; x &= 5;

a) 0
b) 5
c) 10
d) 15

viii. What is the purpose of the sizeof operator in C?

a) It returns the size of a variable or data type in bytes.
b) It returns the number of elements in an array.
c) It returns the address of a variable.
d) It returns the value of a variable.

ix. Which of the following is a valid C type qualifier?

a) extern
b) register
c) static
d) volatile

Expected Outcomes

i. What is the expected output of the following C program?

```
#include <stdio.h>

int main() {
int x = 5;
printf("The value of x is %d", x);
return 0;
}
```

b) The value of x is 5
  b) x
  c) The value of x is undefined
  d) The program will not compile

i. What is the expected output of the following C program?

```
#include <stdio.h>

int main() {
int x = 5;
int y = 7;
printf("The sum of x and y is %d", x + y);
return 0;
}
```

a) The sum of x and y is 12
b) The sum of x and y is 57
c) The program will not compile
d) The sum of x and y is undefined

ii. What is the expected output of the following C program?

```
#include <stdio.h>

int main() {
float x = 3.14;
printf("The value of x is %f", x);
return 0;
}
```

a) The value of x is 3.140000
b) The value of x is 3.14
c) The program will not compile
d) The value of x is undefined

iii. What is the expected output of the following C program?

```
#include <stdio.h>

int main() {
int x = 5;
printf("The value of x is %d\n", x);
x = 7;
printf("The value of x is now %d", x);
return 0;
}
```

a) The value of x is 5, The value of x is now 7
b) The value of x is 5\nThe value of x is now 7
c) The value of x is now 7, The value of x is 5
d) The program will not compile

iv. What is the expected output of the following C program?

```
#include <stdio.h>

int main() {
int x = 5;
int y = 7;
if (x < y) {
printf("x is less than y");
} else {
printf("x is greater than or equal to y");
}
return 0;
}
```

a) x is less than y
b) x is greater than or equal to y
c) The program will not compile
d) x is less than y\nx is greater than or equal to y

v. What is the expected output of the following C program?

```
#include <stdio.h>

int main() {
int x = 5;
x += 3;
printf("The value of x is %d", x);
return 0;
}
```

a) The value of x is 8
b) The value of x is 3
c) The program will not compile
d) The value of x is undefined

vi. What is the expected output of the following C program?

```
#include <stdio.h>

int main() {
int x = 5;
int y = 7;
if (x > y) {
printf("x is greater than y");
} else if (x < y) {
printf("x is less than y");
} else {
printf("x is equal to y");
}
return 0;
  }
```

a) x is less than y
b) x is greater than y
c) x is equal to y
d) The program will not compile

vii. What is the expected output of the following C program?

```
#include <stdio.h>

int main() {
int x = 5;
int y = 7;
int z = x & y;
printf("The value of z is %d", z);
return 0;
}
```

a) The value of z is 1
b) The value of z is 4
c) The program will not compile
d) The value of z is undefined

viii. What is the expected output of the following C program?

```
#include <stdio.h>

int main() {
int x = 5;
int y = 7;
int z = x ^ y;
printf("The value of z is %d", z);
return 0;
}
```

a) The value of z is 2
b) The value of z is 12
c) The program will not compile

d) The value of z is undefined

ix. What is the expected output of the following C program?

```
#include <stdio.h>

int main() {
int x = 5;
int y = 7;
int z = x | y;
printf("The value of z is %d", z);
return 0;
}
```

a) The value of z is 7
b) The value of z is 12
c) The program will not compile
d) The value of z is undefined

UNIT 2 Control Structure and Input/Output functions

Easy

a) Which control structure is used to execute a block of code repeatedly until a specific condition is met?

a) if-else statement
b) while loop
c) switch case statement
d) do-while loop

b) Which control structure is used to execute a block of code only when a specific condition is true?

a) for loop
b) while loop
c) if-else statement
d) do-while loop

c) Which control structure is used to execute a block of code multiple times, with each iteration dependent on the previous iteration's result?

a) if-else statement
b) while loop
c) for loop
d) do-while loop

d) Which control structure is used to jump to a specific labeled statement in the program?

a) break statement
b) continue statement
c) goto statement
d) return statement

e) Which control structure is used to terminate the current iteration of a loop and continue with the next iteration?

a) break statement
b) continue statement
c) goto statement
d) return statement

f) Which control structure is used to exit a loop or switch case statement immediately?

a) break statement
b) continue statement
c) goto statement
d) return statement

g) Which control structure is used to return a value from a function to the calling program?

a) break statement
b) continue statement
c) goto statement
d) return statement

h) Which function is used to read formatted input from the standard input stream in C?

a) scanf()
b) gets()
c) puts()
d) printf()

i) Which function is used to write formatted output to the standard output stream in C?

a) scanf()
b) gets()
c) puts()
d) printf()

j) Which modifier is used to change the storage size of a variable in C?

a) const

b) static
c) extern
d) sizeof

Medium

a) Which control structure is more suitable for situations where the number of iterations is unknown?
a) for loop
b) while loop
c) do-while loop
d) switch case statement

b) Which control structure is used to execute a block of code based on multiple possible outcomes of a single expression?
a) if-else statement
b) while loop
c) for loop
d) switch case statement

c) Which control structure is used to execute a block of code based on multiple possible outcomes of different expressions?
a) if-else statement
b) while loop
c) for loop
d) switch case statement

d) Which control structure is used to execute a block of code at least once, regardless of whether the condition is true or false?
a) if-else statement
b) while loop
c) for loop
d) do-while loop

e) Which control structure is used to jump to a specific labeled statement in the program, and is generally considered bad practice?
a) break statement
b) continue statement
c) goto statement
d) return statement

f) Which modifier is used to prevent a variable from being modified in a program?
a) const
b) static

c) extern
d) volatile

g) Which modifier is used to define a variable that is accessible only within the current block of code?
a) const
b) static
c) extern
d) volatile

h) Which function is used to read unformatted input from the standard input stream in C?
a) scanf()
b) gets()
c) puts()
d) getchar()

i) Which function is used to write unformatted output to the standard output stream in C?
a) scanf()
b) puts()
c) getchar()
d) putchar()

j) Which control structure is used to iterate over each element of an array or sequence?
a) for loop
b) while loop
c) do-while loop
d) switch case statement

Hard

1. What is the output of the following code snippet?
2. css
3. Copy code
4. int a = 5, b = 10;
5. if (a > b) {
6. printf("a is greater than b");
7. } else if (a < b) {
8. printf("b is greater than a");
9. } else {
10. printf("a is equal to b");
11. }
    a) a is greater than b
    b) b is greater than a
    c) a is equal to b

d) Compile error

12. Which of the following is not a valid data type in C?
   a) char
   b) float
   c) double
   d) decimal

13. What is the output of the following code snippet?

14. perl
15. Copy code
16. int i = 0;
17. while (i < 5) {
18. printf("%d\n", i);
19. i++;
20. }
   a) 0 1 2 3 4
   b) 1 2 3 4 5
   c) 0 1 2 3 4 5
   d) Compile error

21. What is the output of the following code snippet?
22. perl
23. Copy code
24. int i = 0;
25. do {
26. printf("%d\n", i);
27. i++;
28. } while (i < 5);
   a) 0 1 2 3 4
   b) 1 2 3 4 5
   c) 0 1 2 3 4 5
   d) Compile error

29. Which control structure is used to execute a block of code based on a condition that is evaluated at the end of each iteration?
   a) for loop
   b) while loop
   c) do-while loop
   d) switch case statement

30. What is the output of the following code snippet?

31. css
32. Copy code
33. int i = 0;
34. for (i = 0; i < 5; i++) {
35. if (i == 3) {
36. break;
37. }
38. printf("%d\n", i);
39. }
    a) 0 1 2
    b) 0 1 2 3
    c) 0 1 2 4
    d) Compile error

40. What is the output of the following code snippet?
41. css
42. Copy code
43. int i = 0;
44. for (i = 0; i < 5; i++) {
45. if (i == 3) {
46. continue;
47. }
48. printf("%d\n", i);
49. }
    a) 0 1 2 4
    b) 0 1 2 3 4
    c) 0 1 2 3
    d) Compile error

50. Which control structure is used to execute a block of code based on a variable that can take on multiple possible values?
    a) if-else statement
    b) while loop
    c) for loop
    d) switch case statement

51. Which modifier is used to define a variable that is accessible across multiple source files in a C program?
    a) const
    b) static
    c) extern
    d) volatile

52. What is the output of the following code snippet?

```
int a = 5, b = 10;
a > b ? printf("a is greater than b") : printf("b is greater than a");
```

a) a is greater than b
b) b is greater than a
c) Compile error
d) Undefined behavior

Expected Outcome

53. What is the expected output of the following code snippet?

```
int i = 1;
while (i <= 10) {
printf("%d ", i);
i++;
}
```

a) 1 2 3 4 5 6 7 8 9 10
b) 1 2 3 4 5 6 7 8 9
c) 2 4 6 8 10
d) 10 9 8 7 6 5 4 3 2 1

54. What is the expected output of the following code snippet?

```
int i;
for (i = 0; i < 5; i++) {
if (i == 3) {
continue;
}
printf("%d ", i);
}
```

a) 0 1 2 4
b) 0 1 2 3 4
c) 1 2 3 4
d) 0 1 2 3

55. What is the expected output of the following code snippet?

```
int i;
for (i = 1; i <= 10; i++) {
if (i % 2 == 0) {
```

```
printf("%d ", i);
}
}
```

a) 2 4 6 8 10
b) 1 3 5 7 9
c) 1 2 3 4 5 6 7 8 9 10
d) 0 2 4 6 8 10

56. What is the expected output of the following code snippet?

```
int x = 2;
switch (x) {
case 1:
printf("One");
break;
case 2:
printf("Two");
break;
case 3:
printf("Three");
break;
default:
printf("Other");
}
```

a) One
b) Two
c) Three
d) Other

57. What is the expected output of the following code snippet?

```
int i;
for (i = 1; i <= 10; i++) {
if (i == 5) {
break;
}
printf("%d ", i);
}
```

a) 1 2 3 4
b) 1 2 3 4 5
c) 1 2 3 4 5 6 7 8 9 10

d) 5 6 7 8 9 10

58. What is the expected output of the following code snippet?

```
int a = 5, b = 10;
if (a > b) {
printf("a is greater than b");
} else {
printf("a is not greater than b");
}
```

a) a is greater than b
b) a is not greater than b
c) Compile error
d) Undefined behavior

59. What is the expected output of the following code snippet?

```
int i;
for (i = 0; i < 5; i++) {
if (i == 2) {
continue;
}
printf("%d ", i);
if (i == 3) {
break;
}
}
```

a) 0 1 3
b) 0 1 2 3
c) 0 1 2 3 4
d) 1 2 3 4

60. What is the expected output of the following code snippet?

```
int i = 1;
do {
printf("%d ", i);
i++;
} while (i <= 10);
```

a) 1 2 3 4 5 6 7 8 9 10
b) 2 4 6 8 10

c) 1
d) Infinite loop

61. What is the expected output of the following code snippet?

```
int a = 10;
int *p = &a;
printf("%d", *p);
```

a) 10
b) 0
c) Compile error
d) Undefined behavior

62. What is the expected output of the following code snippet?

```
int a = 10;
float b = (float)a / 3;
printf("%f", b);
```

a) 3.000000
b) 3.333333
c) 3.333334
d) 10.000000

63. What is the expected output of the following code snippet?

```
char name[20];
printf("Enter your name: ");
scanf("%s", name);
printf("Hello, %s!", name);
```

a) Enter your name: Hello, [input name]!
b) Hello, Enter your name: [input name]!
c) Enter your name: [input name]! Hello,
d) None of the above

64. What is the expected output of the following code snippet?

```
char str[20] = "Hello";
str[4] = '\0';
printf("%s", str);
```

a) Hello
b) Hell

- c) H
- d) Compile error

65. What is the expected output of the following code snippet?

```
int i;
for (i = 0; i < 5; i++) {
printf("%d ", i);
}
```

- a) 0 1 2 3 4
- b) 1 2 3 4 5
- c) 1 3 5
- d) 0 2 4

UNIT 3 User Defined functions and Storage classes

Easy

1. Which of the following storage classes cannot be used for global variables?
   - a) auto
   - b) extern
   - c) static
   - d) register

2. Which of the following storage classes allows a variable to be accessed from multiple source files?
   - a) auto
   - b) extern
   - c) static
   - d) register

3. Which storage class allocates memory on the stack?
   - a) auto
   - b) extern
   - c) static
   - d) register

4. Which of the following storage classes allocates memory in the data segment?
   - a) auto
   - b) extern
   - c) static
   - d) register

5. What is the purpose of a function prototype?

    a) To declare the return type of the function
    b) To declare the function name and parameters
    c) To define the function body
    d) To declare the storage class of the function

6. What is the difference between passing arguments by value and passing arguments by reference?
    a) Passing by value makes a copy of the argument, while passing by reference does not
    b) Passing by reference makes a copy of the argument, while passing by value does not
    c) There is no difference between passing by value and passing by reference
    d) Passing by value and passing by reference are not valid ways to pass arguments to a function

7. What is a recursive function?
    a) A function that calls itself
    b) A function that is called by another function
    c) A function that returns a value
    d) A function that takes arguments by reference

8. What is the scope of a global variable?
    a) The entire program
    b) The function in which it is declared
    c) The block in which it is declared
    d) The parameter list of the function in which it is declared

9. Which header file contains the math library functions?
    a) <stdio.h>
    b) <stdlib.h>
    c) <math.h>
    d) <string.h>

10. What is the purpose of the static storage class?
    a) To allocate memory on the stack
    b) To allocate memory in the data segment
    c) To restrict the scope of a variable to the block in which it is declared
    d) To allow a variable to be accessed from multiple source files

Medium

1. What is the difference between a function prototype and a function definition?
    a) A function prototype declares the function name and parameters, while a function definition contains the function body
    b) A function definition declares the function name and parameters, while a function prototype contains the function body

    c) There is no difference between a function prototype and a function definition
    d) A function prototype is used for recursive functions, while a function definition is used for non-recursive functions

2. Which of the following is a valid way to declare a function that returns an integer and takes two arguments, both of which are integers?
    a) int function(int a, int b);
    b) function(int a, int b) int;
    c) function(int a, int b) { return int; }
    d) int function(int, int);

3. What is the purpose of passing arguments by reference?
    a) To conserve memory by not making a copy of the argument
    b) To create a new copy of the argument
    c) To avoid modifying the original argument
    d) To pass the argument to another function

4. Which of the following is a valid way to declare a recursive function that returns a float and takes one argument, an integer?
    a) float function(int a) { return a; }
    b) function(float a, int b) { return b; }
    c) float function(int a);
    d) function(int a);

5. What is the purpose of the extern storage class?
    a) To allocate memory on the stack
    b) To allocate memory in the data segment
    c) To declare a variable that is defined in another source file
    d) To declare a variable that can only be accessed from within a single function

6. What is the scope of a local variable?
    a) The entire program
    b) The function in which it is declared
    c) The block in which it is declared
    d) The parameter list of the function in which it is declared

7. Which of the following is a math library function that returns the absolute value of a number?
    a) pow()
    b) sqrt()
    c) ceil()
    d) abs()

8. What is the purpose of the register storage class?
    a) To allocate memory on the stack
    b) To allocate memory in the data segment

   c) To optimize access to frequently used variables
   d) To restrict the scope of a variable to the block in which it is declared

9. What is the difference between a global variable and a static variable?
   a) A global variable can only be accessed from within the function in which it is declared, while a static variable can be accessed from anywhere in the program
   b) A global variable is initialized to zero, while a static variable is not initialized
   c) A global variable is stored on the heap, while a static variable is stored on the stack
   d) A global variable has a longer lifetime than a static variable

10. Which of the following is a valid way to define a function that takes no arguments and returns nothing?
    a) function()
    b) void function()
    c) int function()
    d) function(void)

Hard

1. What is the difference between a function pointer and a function?
   a) A function pointer is a variable that points to the address of a function, while a function is a block of code that performs a specific task
   b) A function pointer is a special type of function that can be called using a different syntax than a regular function
   c) A function pointer is a function that returns a pointer to another function
   d) There is no difference between a function pointer and a function

2. What is the output of the following code snippet?

```
#include <stdio.h>
int main() {
int a = 10;
printf("%d\n", ++a + a++);
return 0;
}
```

   a) 20
   b) 21
   c) 22
   d) Undefined behavior

3. What is the difference between a static function and a non-static function?
   a) A static function can only be called from within the file in which it is declared, while a non-static function can be called from any file
   b) A static function has a longer lifetime than a non-static function

    c) A static function is stored on the heap, while a non-static function is stored on the stack
    d) There is no difference between a static function and a non-static function

4. What is the purpose of passing arguments by const reference?
    a) To conserve memory by not making a copy of the argument
    b) To create a new copy of the argument
    c) To avoid modifying the original argument
    d) To pass the argument to another function

5. Which of the following is a math library function that calculates the sine of an angle in radians?
    a) tan()
    b) cosh()
    c) asin()
    d) sin()

6. What is the output of the following code snippet?

```
#include <stdio.h>
int function(int n) {
if(n == 0) {
return 0;
}
else {
return n + function(n - 1);
}
}
int main() {
printf("%d\n", function(5));
return 0;
}
```

    a) 5
    b) 15
    c) 25
    d) 120

7. What is the difference between a static variable and a constant variable?
    a) A static variable can be modified, while a constant variable cannot be modified
    b) A static variable has a longer lifetime than a constant variable
    c) A static variable can only be accessed from within the function in which it is declared, while a constant variable can be accessed from anywhere in the program
    d) There is no difference between a static variable and a constant variable

8. Which of the following is a valid way to define a function that takes a variable number of arguments?
    a) void function(...) {}
    b) void function(int a, ...) {}
    c) int function(...) { return 0; }
    d) function(int a, ...) { return 0; }

9. What is the purpose of the auto storage class?
    a) To allocate memory on the stack
    b) To allocate memory in the data segment
    c) To optimize access to frequently used variables
    d) To restrict the scope of a variable to the block in which it is declared

10. What is the output of the following code snippet?

```
#include <stdio.h>
int x = 10;
void function() {
static int x = 5;
x++;
printf("%d\n", x);
}
int main() {
int x = 20;
function();
function();
function();
printf("%d\n", x);
printf("%d\n", ::x);
return 0;
```

    a) 6 7 8 20 10
    b) 5 6 7 20 10
    c) 6 7 8 10 20
    d) 5 6 7 10 20

Expected Outcomes

1. What is the expected output of the following code snippet?

```
#include <stdio.h>
#include <math.h>
int main() {
double x = 2.0;
double y = sqrt(x);
printf("%f\n", y);
return 0;
}
```

a) 1.414214
b) 2.000000
c) 3.141593
d) 4.000000

2. What is the expected output of the following code snippet?

```
#include <stdio.h>
void function(int x, int y) {
int z = x + y;
printf("%d\n", z);
}
int main() {
function(5, 10);
return 0;
}
```

a) 5
b) 10
c) 15
d) Undefined behavior

3. What is the expected output of the following code snippet?

```
#include <stdio.h>
int main() {
int x = 5;
if(x == 5) {
int y = 10;
printf("%d\n", y);
}
printf("%d\n", y);
return 0;
}
```

a) 10
b) 5
c) Undefined behavior
d) Compilation error

4. What is the expected output of the following code snippet?

```
#include <stdio.h>
void function(int *x) {
```

```
*x = 10;
}
int main() {
int x = 5;
function(&x);
printf("%d\n", x);
return 0;
}
```

5.

a) 5
b) 10
c) Undefined behavior
d) Compilation error

6. What is the expected output of the following code snippet?

```
#include <stdio.h>
int main() {
int x = 5;
{
int x = 10;
printf("%d\n", x);
}
printf("%d\n", x);
return 0;
}
```

7.

a) 5 10
b) 10 5
c) 10 10
d) Compilation error

8. What is the expected output of the following code snippet?

```
#include <stdio.h>
int function(int n) {
if(n == 0) {
return 0;
}
else {
return n + function(n - 1);
}
}
int main() {
printf("%d\n", function(3));
return 0;
}
```

a) 3
b) 6
c) 9
d) 10

9. What is the expected output of the following code snippet?

```
#include <stdio.h>
void function() {
static int x = 5;
x++;
printf("%d\n", x);
}
int main() {
function();
function();
function();
return 0;
}
```

a) 6 7 8
b) 5 6 7
c) 6 6 6
d) 5 5 5

10. What is the expected output of the following code snippet?

```
#include <stdio.h>
void function(int x) {
static int y = 0;
y += x;
printf("%d\n", y);
}
int main() {
function(5);
function(10);
function(15);
return 0;
}
```

a) 5 15 30
b) 5 10 15
c) 5 10 25

d) 5 15

UNIT 4 Pointers & Arrays in C

Easy

1. What is a pointer in C programming?
    a) A variable that stores the address of another variable
    b) A variable that stores the value of another variable
    c) A variable that stores a character value
    d) A variable that stores an integer value

2. How do you declare a pointer variable in C?
    a) int *ptr;
    b) char *ptr;
    c) float *ptr;
    d) All of the above

3. Which operator is used to access the value at the address stored in a pointer variable?
    a) &
    b) *
    c) %
    d) /

4. What does the following expression mean: ptr++;
    a) Increment the value of the pointer by 1
    b) Decrement the value of the pointer by 1
    c) Access the next element of an array
    d) None of the above

5. What is the purpose of passing a pointer to a function?

   a) To modify the value of the original variable
   b) To avoid passing a large data structure
   c) To reduce the number of arguments passed to the function
   d) All of the above

6. How do you declare a one-dimensional array in C?
   a) int array[5];
   b) char array[10];
   c) float array[3];
   d) All of the above

7. Which of the following is an example of a two-dimensional array?
   a) int array[5];
   b) char array[10][5];
   c) float array[3][2];
   d) None of the above

8. What is the process of sorting an array called in C?
   a) Shuffling
   b) Merging
   c) Sorting
   d) Reversing

9. Which of the following is a linear search algorithm?
   a) Binary search
   b) Selection sort
   c) Bubble sort
   d) Linear search

10. What is a null pointer in C?
   a) A pointer that points to a random memory location
   b) A pointer that points to the first element of an array
   c) A pointer that points to nothing
   d) A pointer that points to the last element of an array.

Medium

1. What is the difference between a pointer and an array in C?
    a) An array can only store values of the same data type, while a pointer can store different data types
    b) An array is a collection of contiguous memory locations, while a pointer is a variable that stores the address of a memory location
    c) An array can be resized dynamically, while a pointer has a fixed size
    d) An array is used to store multiple values, while a pointer is used to manipulate memory addresses

2. How do you initialize a pointer variable in C?
    a) int *ptr = 5;
    b) int *ptr = #
    c) int *ptr = &*num;
    d) None of the above

3. Which of the following is an example of pointer arithmetic in C?
    a) ptr + 1
    b) ptr - 1
    c) ptr++
    d) All of the above

4. What is the purpose of a void pointer in C?
    a) To store a memory address of any data type
    b) To store a null value
    c) To store a constant value
    d) None of the above

5. How do you pass an array to a function in C?
    a) By passing the array name
    b) By passing the address of the first element of the array

    c) By passing the value of the first element of the array
    d) By passing the size of the array

6. What is the purpose of a multidimensional array in C?
    a) To store multiple values of the same data type
    b) To store multiple values of different data types
    c) To represent complex data structures
    d) None of the above

7. How do you insert an element into an array in C?
    a) By moving all the elements to the right and inserting the new element in the empty space
    b) By overwriting an existing element with the new value
    c) By appending the new element to the end of the array
    d) None of the above

8. What is the purpose of a binary search algorithm?
    a) To sort an array in ascending order
    b) To search for a value in an array in O(log n) time
    c) To count the number of elements in an array
    d) None of the above

9. How do you delete an element from an array in C?
    a) By moving all the elements to the left and deleting the last element
    b) By overwriting the element with a null value
    c) By shifting all the elements after the deleted element to the left
    d) None of the above

10. What is the difference between bubble sort and insertion sort?
    a) Bubble sort is faster than insertion sort
    b) Bubble sort compares adjacent elements, while insertion sort inserts elements in their correct position
    c) Bubble sort is only used for sorting arrays of small sizes, while insertion sort can be used for any size of array

    d) None of the above

Hard

1. What is the purpose of a dangling pointer in C?
    a) To point to a valid memory address
    b) To point to a memory address that has been deallocated
    c) To point to a null value
    d) None of the above

2. How do you declare a pointer to a function in C?
    a) int (*ptr)(int);
    b) int *ptr(int);
    c) int ptr(*int);
    d) None of the above

3. What is the purpose of a wild pointer in C?
    a) To point to a valid memory address
    b) To point to a memory address that has been deallocated
    c) To point to a random memory address
    d) None of the above

4. How do you declare an array of pointers in C?
    a) int *arr[];
    b) int **arr;
    c) int *arr[10];
    d) None of the above

5. What is the purpose of a generic pointer in C?
    a) To point to a memory address of any data type
    b) To point to a null value
    c) To point to a constant value
    d) None of the above

6. What is the time complexity of bubble sort?
    a) O(n)
    b) O(n^2)
    c) O(n log n)
    d) O(log n)

7. What is the purpose of a double pointer in C?
    a) To store the address of a pointer variable
    b) To store the address of an array
    c) To create a pointer to a pointer
    d) None of the above

8. How do you pass a multidimensional array to a function in C?
    a) By passing the array name
    b) By passing the address of the first element of the array
    c) By passing the value of the first element of the array
    d) By passing the size of the array

9. What is the purpose of a null pointer in C?
    a) To point to a valid memory address
    b) To point to a memory address that has been deallocated
    c) To point to a null value
    d) None of the above

10. How do you perform a binary search on a multidimensional array in C?
    a) By flattening the array and performing a binary search
    b) By performing a binary search on each subarray separately
    c) By performing a linear search on each subarray separately
    d) None of the above

Expected Outcome

1. What is the expected output of the following code?

```
int x = 10;

int *p;

p = &x;

printf("%d", *p);
```

   a) 10
   b) 0
   c) 1
   d) None of the above

2. What is the expected output of the following code?

```
int arr[5] = {1, 2, 3, 4, 5};

int *p = arr;

printf("%d", *(p + 3));
```

   a) 1
   b) 2
   c) 4
   d) 5

3. What is the expected output of the following code?

```
int arr[3][3] = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};

int *p = arr[0];

printf("%d", *(p + 5));
```

   a) 1
   b) 5

c) 6
d) 8

4. What is the expected output of the following code?

```
int arr[5] = {5, 3, 1, 4, 2};

int n = sizeof(arr)/sizeof(int);

bubbleSort(arr, n);

for(int i = 0; i < n; i++) {

printf("%d ", arr[i]);

}
```

a) 1 2 3 4 5
b) 5 4 3 2 1
c) 3 1 4 2 5
d) None of the above

5. What is the expected output of the following code?

```
int arr[5] = {1, 2, 3, 4, 5};

reverse(arr, 5);

for(int i = 0; i < 5; i++) {

printf("%d ", arr[i]);

}
```

a) 1 2 3 4 5
b) 5 4 3 2 1
c) 4 3 2 1 5
d) None of the above

6. What is the expected output of the following code?

```
int arr[3][3] = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};

int n = sizeof(arr)/sizeof(arr[0]);

int m = sizeof(arr[0])/sizeof(int);

int k = binarySearch(arr, n, m, 5);

printf("%d", k);
```

a) 0
b) 1
c) 4
d) None of the above

7. What is the expected output of the following code?

```
int arr[3][3] = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};

int *p[3] = {arr[0], arr[1], arr[2]};

int n = sizeof(p)/sizeof(int *);

selectionSort(p, n);

for(int i = 0; i < n; i++) {

for(int j = 0; j < 3; j++) {

printf("%d ", *(p[i] + j));

}

}
```

a) 1 2 3 4 5 6 7 8 9
b) 7 8 9 4 5 6 1 2 3
c) 1 4 7 2 5 8 3

8. What is the expected output of the following code?

```
int main() {

int a[3] = {1, 2, 3};
int *p = a;
printf("%d", *p + 2);
return 0;
}
```

a) 1
b) 2
c) 3
d) 5

9. What is the expected output of the following code?

```
int main() {
int a[3] = {1, 2, 3};
int *p = a;
printf("%d", *(p+2));
return 0;
}
```

a) 1
b) 2
c) 3
d) garbage value

10. What is the expected output of the following code?

```
int main() {
int a[3] = {1, 2, 3};
int *p = &a[1];
printf("%d", *(p-1));

return 0;

}
```

a) 1
b) 2
c) 3
d) garbage value

11. What is the expected output of the following code?

```
int main() {

int a[3][3] = {{1,2,3},{4,5,6},{7,8,9}};

int (*p)[3] = a;

printf("%d", *(*p+2));

return 0;

}
```

a) 1
b) 2
c) 3
d) 4

12. What is the expected output of the following code?

```
int main() {

int a[3] = {1, 2, 3};

int *p = &a[0];

int *q = &a[2];

printf("%d", q-p);

return 0;
}
```

a) 0
b) 1
c) 2
d) 3

UNIT 5 Dynamic Memory Management & File I/O

Easy

i) Which function is used to allocate memory dynamically in C language?
   a. malloc()
   b. calloc()
   c. realloc()
   d. All of the above

ii) Which function is used to free the memory allocated by the malloc() function?
   a. free()
   b. calloc()
   c. realloc()
   d. None of the above

iii) What happens if we try to access a pointer that has already been freed?
   a. The program crashes
   b. The pointer becomes null
   c. The pointer still points to the same memory location
   d. None of the above

iv) Which function is used to allocate memory for an array of elements in C language?
   a. malloc()
   b. calloc()
   c. realloc()
   d. None of the above

v) Which of the following is an example of a memory leak?
   a. Forgetting to close a file after reading it

b. Allocating memory but not freeing it
c. Using uninitialized variables
d. None of the above

vi) Which of the following is used to open a file in C language?

a. fopen()
b. fclose()
c. fprint()
d. None of the above

vii) Which of the following is used to read data from a file in C language?

a. fread()
b. fwrite()
c. fputc()
d. None of the above

viii) Which of the following is used to write data to a file in C language?

a. fread()
b. fwrite()
c. fgetc()
d. None of the above

ix) Which of the following is used to append data to a file in C language?

a. fseek()
b. ftell()
c. fprintf()
d. None of the above

x) Which of the following is used to perform random access of a file in C language?

a. fseek()
b. ftell()
c. fprint()
d. None of the above

Medium

i) What is dynamic memory allocation?

a. Allocating memory at compile-time
b. Allocating memory at run-time
c. Allocating memory in the heap

d. None of the above

ii) What is the difference between malloc() and calloc() functions?

a. malloc() allocates memory for a single element, while calloc() allocates memory for multiple elements
b. malloc() initializes the allocated memory to zero, while calloc() does not
c. calloc() initializes the allocated memory to zero, while malloc() does not
d. None of the above

iii) What is the purpose of the realloc() function?

a. To allocate memory dynamically
b. To free memory dynamically
c. To reallocate memory dynamically
d. None of the above

iv) What happens if realloc() fails to allocate memory?

a. It returns a null pointer
b. It returns the original pointer
c. It causes the program to crash
d. None of the above

v) What is a memory leak?

a. When memory is not freed after use
b. When memory is not allocated
c. When memory is allocated but not used
d. None of the above

vi) How can memory leaks be detected?

a. By analyzing the code
b. By using a memory profiler tool
c. By running the program and observing its behavior
d. None of the above

vii) What is the purpose of the FILE structure in C language?

a. To store file data
b. To store file metadata
c. To store both file data and metadata
d. None of the above

viii) What is the difference between text and binary files?

  a. Text files can only store text, while binary files can store any type of data
  b. Text files are human-readable, while binary files are not
  c. Text files are smaller in size than binary files
  d. None of the above

ix) What is the purpose of fseek() function in C language?

  a. To move the file pointer to a specific location in a file
  b. To read data from a file
  c. To write data to a file
  d. None of the above

x) What is the purpose of ftell() function in C language?

  a. To get the current position of the file pointer in a file
  b. To get the size of a file in bytes
  c. To get the last error that occurred on a file stream
  d. None of the above

Hard

i) What is a segmentation fault in C language?

  a. An error caused by memory allocation failure
  b. An error caused by accessing memory that is not allocated or is already freed
  c. An error caused by a syntax error in the code
  d. None of the above

ii) How can you avoid memory leaks in C language?

  a. By always freeing allocated memory after use
  b. By using garbage collection
  c. By avoiding dynamic memory allocation altogether
  d. None of the above

iii) What is the difference between stack and heap memory in C language?

  a. Stack memory is allocated at compile-time, while heap memory is allocated at run-time
  b. Stack memory is limited in size, while heap memory is not

c. Stack memory is automatically managed by the compiler, while heap memory is managed by the programmer
d. None of the above

iv) What is the purpose of the valgrind tool in C language?
a. To detect memory leaks in C programs
b. To profile the performance of C programs
c. To optimize the memory usage of C programs
d. None of the above

v) What is the purpose of the mmap() function in C language?
a. To allocate memory in the stack
b. To allocate memory in the heap
c. To map a file into memory
d. None of the above

vi) What is the difference between fread() and fgets() functions in C language?
a. fread() reads a binary file, while fgets() reads a text file
b. fread() reads a fixed number of bytes from a file, while fgets() reads a line of text from a file
c. fgets() reads a fixed number of bytes from a file, while fread() reads a line of text from a file
d. None of the above

vii) What is the difference between fprintf() and fputs() functions in C language?
a. fprintf() writes formatted data to a file, while fputs() writes unformatted data to a file
b. fprintf() writes unformatted data to a file, while fputs() writes formatted data to a file
c. fprintf() and fputs() are equivalent functions
d. None of the above

viii) What is the difference between binary and text mode when opening a file in C language?
a. Binary mode can only be used for reading binary files, while text mode can only be used for reading text files
b. In binary mode, data is read and written in its raw form, while in text mode, data is converted to and from its textual representation

c. In text mode, data is read and written in its raw form, while in binary mode, data is converted to and from its binary representation
d. None of the above

ix) What is the purpose of the setbuf() function in C language?

a. To set the buffering mode for a file
b. To read data from a file
c. To write data to a file
d. None of the above

x) What is the purpose of the fseek() function with SEEK_END option in C language?

a. To move the file pointer to the beginning of a file
b. To move the file pointer to the end of a file
c. To move the file pointer to a specific location in a file
d. None of the above

Expected Outcome

i) What is the expected output of the following code segment?

```
int* ptr = (int*) malloc(sizeof(int));

*ptr = 42;

free(ptr);

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

a. 42
b. 0
c. undefined behavior
d. compilation error

ii) What is the expected output of the following code segment?

```
FILE* fp = fopen("file.txt", "r");

char str[100];

fscanf(fp, "%s", str);

printf("%s\n", str);
```

```
fclose(fp);
```

Assuming that the file "file.txt" contains the text "Hello, world!"

a. Hello, world!
b. Hello,
c. world!
d. compilation error

iii) What is the expected output of the following code segment?

```
int* arr = (int*) calloc(5, sizeof(int));

for(int i=0; i<5; i++){

    printf("%d ", arr[i]);

}

free(arr);
```

a. 0 0 0 0 0
b. 0 1 2 3 4
c. undefined behavior
d. compilation error

iv) What is the expected output of the following code segment?

```
FILE* fp = fopen("file.txt", "w");

char str[100] = "Hello, world!";

fprintf(fp, "%s", str);

fclose(fp);
```

Assuming that the file "file.txt" did not exist before running the code.

a. The file "file.txt" is created and contains the text "Hello, world!"
b. The file "file.txt" is created but it is empty
c. undefined behavior
d. compilation error

v) What is the expected output of the following code segment?

```
int* arr = (int*) malloc(3 * sizeof(int));
```

```
for(int i=0; i<5; i++){
    arr[i] = i;
    printf("%d ", arr[i]);
}
free(arr);
```

a. 0 1 2 0 0
b. 0 1 2
c. undefined behavior
d. compilation error

vi) What is the expected output of the following code segment?

```
FILE* fp = fopen("file.bin", "wb");
int arr[3] = {1, 2, 3};
fwrite(arr, sizeof(int), 3, fp);
fclose(fp);


fp = fopen("file.bin", "rb");
int buffer[3];
fread(buffer, sizeof(int), 3, fp);
for(int i=0; i<3; i++){
    printf("%d ", buffer[i]);
}
fclose(fp);
```

a. 1 2 3
b. 3 2 1
c. undefined behavior
d. compilation error

vii) What is the expected output of the following code segment?

```
int* ptr = (int*) malloc(sizeof(int));
*ptr = 42;
printf("%p\n", ptr);
ptr = realloc(ptr, 2*sizeof(int));
printf("%p\n", ptr);
free(ptr);
```

a. Two memory addresses
b. One memory address
c. undefined behavior
d. compilation error

viii)What is the expected output of the following code segment?

```
FILE* fp = fopen("file.txt", "a");
char str[100] = "Hello, world!";
fprintf(fp, "%s", str);
fclose(fp);

fp = fopen("file.txt", "r");
char buffer[100];
fgets(buffer, 100, fp);
printf("%s\n", buffer);
fclose(fp);
```

Assuming that the file "file.txt" already existed and

User

continue writing

ChatGPT

contains the text "Goodbye!"

a. Goodbye!
b. Hello, world!Goodbye!
c. Goodbye!Hello, world!
d. compilation error

ix) What is the expected output of the following code segment?

```
int* ptr = (int*) malloc(sizeof(int));
*ptr = 42;
int* ptr2 = ptr;
free(ptr2);
printf("%d\n", *ptr);
```

a. 42
b. 0
c. undefined behavior
d. compilation error

x) What is the expected output of the following code segment?

```
FILE* fp = fopen("file.txt", "w");
char str[100] = "Hello, world!";
int n = strlen(str);
for(int i=0; i<n; i++){
    fputc(str[i], fp);
}
fclose(fp);
```

```
fp = fopen("file.txt", "r");

char buffer[100];

fgets(buffer, 100, fp);

printf("%s\n", buffer);

fclose(fp);
```

Assuming that the file "file.txt" did not exist before running the code.

a. The file "file.txt" is created and contains the text "Hello, world!"
b. The file "file.txt" is created but it is empty
c. undefined behavior
d. compilation error

UNIT 6 Strings,Derived types including structures and unions

Easy

1. Which of the following is a derived type?
    a. int
    b. float
    c. structure
    d. double

2. Which function is used to read a string from user input?
    a. scanf()
    b. printf()
    c. gets()
    d. puts()

3. Which of the following is NOT a string manipulation function?
    a. strlen()
    b. strcat()

  c. strcmp()
  d. swap()

4. Which operator is used for character arithmetic?
    a. +
    b. -
    c. *
    d. /

    e. What is the output of the following code snippet?
    f. char str[] = "Hello";
    g. printf("%s", str);

        a) Hello
        b) ello
        c) H
        d) llo

    h. Which library function is used to find the length of a string?
        a) strlen()
        b) strrev()
        c) strcpy()
        d) strupr()

    i. Which of the following is a correct declaration of a structure?
        a) struct myStruct { int x; }
        b) struct myStruct { x = 5; }
        c) struct { int x; } myStruct;
        d) struct myStruct (int x;)

    j. How do you access a member of a structure?
        a) Using the dot (.) operator
        b) Using the arrow (->) operator
        c) Using the asterisk (*) operator
        d) Using the pound (#) operator

    k. What is the purpose of a union?
        a) To group related data items

b) To provide a convenient way to access different data types
c) To define a new data type
d) To create a copy of an existing data type

l. What is the output of the following code snippet?
m. union myUnion {
n. int x;
o. float y;
p. };
q. printf("%d", sizeof(myUnion));

a) 4
b) 8
c) 2
d) 6

Medium

a. Which of the following is the correct way to initialize a string?
   a) char str[5] = "Hello";
   b) char str[] = "Hello";
   c) char str[5] = {'H', 'e', 'l', 'l', 'o'};
   d) char str = "Hello";

b. Which function is used to write a string to a file?
   a) scanf()
   b) printf()
   c) fgetc()
   d) fputs()

c. Which library function is used to compare two strings?
   a) strcmp()
   b) strrev()
   c) strcat()
   d) strcpy()

d. What is the ASCII value of the character 'A'?
   a) 64

};

- b) 65
- c) 66
- d) 67

e. What is the output of the following code snippet?

```
char str1[] = "Hello";

char str2[] = "World";

strcat(str1, str2);

printf("%s", str1);
```

- a) HelloWorld
- b) Hello World
- c) WorldHello
- d) HWorldello

f. Which of the following is the correct way to declare a nested structure?

```
struct myStruct { int x; struct nestedStruct { float y; }; }

struct nestedStruct { float y; };
```

a) struct myStruct { int x; nestedStruct ns; }

b) struct myStruct { int x; nestedStruct { float y; }; }

c) struct nestedStruct { float y; };

d) struct myStruct { int x; nestedStruct ns; };

g. What is the output of the following code snippet?

```
struct myStruct {

int x;

float y;

};
```

```
struct myStruct s1 = { 5, 3.14 };
struct myStruct s2 = s1;
printf("%d %f", s2.x, s2.y);
```

a) 5 3.14
b) 3.14 5
c) 5 5
d) 3.14 3.14

h. Which operator is used to access a member of a structure through a pointer?

a) .
b) ->
c) *
d) &

i. What is the output of the following code snippet?

```
union myUnion {
int x;
float y;
};
union myUnion u;
u.x = 10;
printf("%f", u.y);
```

a) 10.0
b) 0.0
c) -10.0
d) Depends on the compiler

j. Which library function is used to convert a string to a long integer?

a) atoi()

    b) atol()
    c) strtol()
    d) strtof()

Hard

a. What is the output of the following code snippet?

```
struct myStruct {

int x;

};

struct myStruct arr[] = { {1}, {2}, {3} };

struct myStruct *ptr = arr;

printf("%d", ptr[1].x);
```

    a) 1
    b) 2
    c) 3
    d) Compilation error

b. Which library function is used to find the length of a string?

    a) strsize()
    b) strlen()
    c) strcount()
    d) strlength()

c. What is the output of the following code snippet?

```
union myUnion {

int x;

float y;

};

union myUnion u;
```

u.y = 10.5;

printf("%d %f", u.x, u.y);

  a) 10 10.5
  b) 1092616192 10.5
  c) 10.5 10
  d) Depends on the compiler

d. Which of the following is the correct way to declare a nested structure?
  a) struct myStruct { int x; struct nestedStruct { float y; } ns; };
  b) struct myStruct { int x; nestedStruct { float y; } ns; };
  c) struct myStruct { int x; struct { float y; } ns; };
  d) struct myStruct { int x; struct nestedStruct { float y; } *ns; };

e. What is the output of the following code snippet?

char str[] = "Hello";

printf("%s", str + 2);

  a) Hel
  b) lo
  c) Hello
  d) Runtime error

f. Which library function is used to find the first occurrence of a substring in a string?
  a) strstr()
  b) strchr()
  c) strrchr()
  d) strspn()

g. What is the output of the following code snippet?

struct myStruct {

int x;

```
};
struct myStruct arr[] = { {1}, {2}, {3} };
printf("%d", arr[1].x);
```

a) 1
b) 2
c) 3
d) Compilation error

h. Which of the following is the correct way to access a member of a structure using a pointer?
   a) ptr->member
   b) ptr.member
   c) *ptr.member
   d) *ptr->member

i. What is the output of the following code snippet?

```
struct myStruct {
int x;
};
struct myStruct *ptr;
ptr->x = 10;
printf("%d", ptr->x);
```

a) 10
b) 0
c) Runtime error
d) Compilation error

j. Which library function is used to concatenate two strings?
   a) strcat()
   b) strncat()

c) strcpy()
d) strncpy()

Expected Outcome

a. What is the output of the following code snippet?

```
char str[] = "Hello World!";

int len = strlen(str);

printf("%d", len);
```

a) 12
b) 11
c) 13
d) 10

b. Which of the following is the correct way to initialize a structure at the time of declaration?

a) struct myStruct s = {1, 2.5};
b) struct myStruct s = {1};
c) struct myStruct s = {.y = 2.5};
d) struct myStruct s; s.x = 1; s.y = 2.5;

c. What is the output of the following code snippet?

```
char str1[] = "Hello";

char str2[] = "World";

strcat(str1, str2);

printf("%s", str1);
```

a) Hello
b) World
c) HelloWorld
d) Runtime error

d. Which of the following library functions is used to compare two strings?

a) strcmp()
b) stricmp()
c) strncmp()
d) strcasecmp()

e. What is the output of the following code snippet?

```
struct myStruct {

int x;

};

struct myStruct *ptr;

ptr = (struct myStruct*)malloc(sizeof(struct myStruct));

ptr->x = 10;

printf("%d", ptr->x);

free(ptr);
```

a) 0
b) 10
c) Runtime error
d) Undefined behavior

f. Which of the following is the correct way to declare a union?

a) union myUnion { int x; char y; };
b) struct myUnion { int x; char y; };
c) myUnion { int x; char y; };
d) union { int x; char y; } myUnion;

g. What is the output of the following code snippet?

```
char str[] = "Hello";

printf("%c", *(str + 1));
```

a) H

  b) e
  c) l
  d) o

h. Which of the following library functions is used to convert a string to a floating point number?
  a) strtof()
  b) atof()
  c) strtod()
  d) None of the above

i. What is the output of the following code snippet?

```
struct myStruct {

int x;

};

struct myStruct arr[] = { {1}, {2}, {3} };

struct myStruct *ptr = arr;

printf("%d", ptr->x);
```

  a) 1
  b) 2
  c) 3
  d) Compilation error

j. Which of the following is the correct way to access a member of a nested structure?
  a) myStruct1.ns1.y
  b) myStruct1.ns2->y
  c) myStruct1.ns2.y
  d) myStruct1.ns1->y