**Time allowed: 90 Minutes Max. Marks: 40**

**General Instructions:**

* **Follow the instructions given in each section.**
* **Make sure that you attempt the questions in order.**

**SECTION-A (10\*1 mark=10 marks)**

***(All questions are compulsory)***

1. What is the return type of the fopen() function?
2. **FILE \***
3. int
4. void
5. char \*
6. Which of the following is true about a pointer to a constant?
   1. The pointer cannot be reassigned to point to a different address
   2. The pointer cannot be dereferenced
   3. **The value it points to cannot be modified**
   4. The pointer itself is constant and cannot be modified
7. In C, what data structure is typically used to implement a stack?
   1. **Array**
   2. Linked List
   3. Queue
   4. Tree
8. What is the concept of a "null pointer" in C++?
   1. **A pointer that points to a memory location with the value 0**
   2. A pointer that points to itself
   3. A pointer that is uninitialized
   4. A pointer that has a value equal to the maximum address of the system
9. Which of the following is a valid declaration for a function that takes a pointer to a pointer as an argument?
   1. **void func(int\*\* ptr);**
   2. void func(int\* \*ptr);
   3. void func(int\* ptr\*);
   4. void func(int\*\* \*ptr);
10. What is the time complexity of a recursive function that divides the problem into two smaller subproblems?
    1. O(1)
    2. **O(log N)**
    3. O(N)
    4. O(N^2)
11. Which of the following is NOT considered while resolving a function overloading in C++?
    1. Number of parameters
    2. Order of parameters
    3. Return type

**d) Function body**

1. Which preprocessor directive is used to conditionally compile a block of code in C++?
   1. #include
   2. #define
   3. **#if**
   4. #ifdef
2. What is the maximum number of lines allowed in an inline function?
   1. 5
   2. 10
   3. **No specific limit**
   4. 20
3. What is the best practice to avoid dangling pointers?
   1. Always initialize pointers to NULL.
   2. Avoid returning pointers to local variables from functions.
   3. Use proper memory deallocation techniques.
   4. **All of the above.**

**SECTION-B (5\*2 mark=10 marks)**

***(All questions are compulsory)***

11) What is the output of the following code snippet?

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

const int\* ptr = arr;

printf("%d", \*ptr++);

**a) 1**

b) 2

c) 3

d) The code results in undefined behavior

12) What does the following code snippet do?

int\*\* ptr;

ptr = nullptr;

**a) Initializes the pointer to nullptr.**

b) Deallocates the memory pointed to by ptr.

c) Sets ptr to point to itself.

d) None of the above.

13) What will be the output of the following code snippet?

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

{

if (i % 2 == 0)

cout << i;

else

break;

}

a) 0

b) 02

**c) 0**

d) Compilation error

14) What would be printed from the following C++ program?

#include <iostream>

#include <stdlib.h>

using namespace std;

int main()

{

float x = 5.999;

float\* y, \*z;

y = &x;

z = y;

cout << x << ", " << \*(&x) << ", " << \*y << ", " << \*z << "\n";

return 0;

}

**a) 5.999, 5.999, 5.999, 5.999**

b) 5.999, 5.9, 5.000, 5.900

c) Address of the elements

d) compilation error

15) What is output of below program?

int main()

{

const int a=10;

a++;

cout<<a;

return 0;

}

a) 10

b) 11

**c) Compilation Error**

d) Linking Error

**SECTION-C(Coding Question) (2x5 marks=5 marks)**

Q16) Nihar is reading the data from files. He wants to read all the data in reverse order. Help him by implementing a program to reverse read the contents of a text file.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Test Case 1** | **Test Case 2** | **Test Case 3** |
| **Input** | File= hello | File= Learn coding | File= Name Roll |
| **Output** | Olleh | gnidoc nraeL | lloR emaN |

Solution :

**#include <stdio.h>**

**int main() {**

**FILE \*file;**

**char ch;**

**long fileSize, i;**

**// Open the file in read mode**

**file = fopen("text.txt", "r");**

**if (file == NULL) {**

**printf("Unable to open the file.\n");**

**return 1;**

**}**

**// Get the file size**

**fseek(file, 0, SEEK\_END);**

**fileSize = ftell(file);**

**// Read and print the file contents in reverse order**

**for (i = fileSize - 1; i >= 0; i--) {**

**fseek(file, i, SEEK\_SET);**

**ch = fgetc(file);**

**printf("%c", ch);**

**}**

**// Close the file**

**fclose(file);**

**return 0;**

**}**

Q17) You are tasked to perform addition on complex numbers. Create a calculator to perform addition of complex numbers using structures.

Each complex number has a real part and an imaginary part.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Test Case 1** | **Test Case 2** | **Test Case 3** |
| **Input** | Enter first complex number:  Real part: 3  Imaginary part: 2  Enter second complex number:  Real part: 1  Imaginary part: 2 | Enter first complex number:  Real part: 4  Imaginary part: 6  Enter second complex number:  Real part: 7  Imaginary part: 9 | Enter first complex number:  Real part: 93  Imaginary part: 2  Enter second complex number:  Real part: 43  Imaginary part: 4 |
| **Output** | Sum of complex numbers: 4.00 + 4.00i | Sum of complex numbers: 11.00 + 15.00i | Sum of complex numbers: 136.00 + 6.00i |

Solution :

**#include <stdio.h>**

**// Structure definition for subject**

**struct Subject {**

**char name[20];**

**int marks;**

**};**

**// Structure definition for student**

**struct Student {**

**char name[50];**

**int rollNo;**

**struct Subject subjects[3];**

**};**

**int main() {**

**// Array of student structures**

**struct Student students[3];**

**// Input student information**

**for (int i = 0; i < 3; i++) {**

**printf("Enter name and roll number for student %d: ", i + 1);**

**scanf("%s %d", students[i].name, &students[i].rollNo);**

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

**printf("Enter marks for subject %d: ", j + 1);**

**scanf("%d", &students[i].subjects[j].marks);**

**}**

**}**

**// Calculate and display average marks for each student**

**printf("\nStudent Database:\n");**

**for (int i = 0; i < 3; i++) {**

**int totalMarks = 0;**

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

**totalMarks += students[i].subjects[j].marks;**

**}**

**float averageMarks = totalMarks / 3.0;**

**printf("Name: %s, Roll Number: %d, Average Marks: %.2f\n",**

**students[i].name, students[i].rollNo, averageMarks);**

**}**

**return 0;**

**}**

**SECTION-D (Coding Question)(1x10 mark=10 mark)**

Q18) User wants to get maximum element from the stack of numbers. Implement a C program to which can get the maximum value in the stack in O(1) time without using an additional stack.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Test Case 1** | **Test Case 2** | **Test Case 3** |
| **Input** | 45 67 89 43 12 | -5 -2 -7 -1 | 0 14 54 389 101 203 |
| **Output** | Maximum value in the stack: 89 | Maximum value in the stack: 1 | Maximum value in the stack: 389 |

Solution :

**#include <stdio.h>**

**#include <stdlib.h>**

**// Structure to represent a block in the stack**

**struct Block {**

**int value; // Value of the block**

**int localMax; // Local maximum value in the stack until this block**

**};**

**// Structure to represent the stack**

**typedef struct {**

**struct Block\* S; // Array of blocks**

**int size; // Maximum size of the stack**

**int top; // Index of the top block in the stack**

**} Stack;**

**// Function to create a stack of a given size**

**Stack\* createStack(int size) {**

**Stack\* stack = (Stack\*)malloc(sizeof(Stack)); // Allocate memory for the stack structure**

**stack->S = (struct Block\*)malloc(size \* sizeof(struct Block)); // Allocate memory for the blocks array**

**stack->size = size; // Set the maximum size of the stack**

**stack->top = -1; // Initialize the top index to -1 (empty stack)**

**return stack;**

**}**

**// Function to push an element onto the stack**

**void push(Stack\* stack, int n) {**

**if (stack->top == stack->size - 1) { // Check if the stack is already full**

**printf("Stack is full\n");**

**}**

**else {**

**stack->top++; // Increment the top index**

**if (stack->top == 0) { // If it's the first element in the stack**

**stack->S[stack->top].value = n; // Set the value of the block**

**stack->S[stack->top].localMax = n; // Set the local maximum as the value itself**

**}**

**else {**

**if (stack->S[stack->top - 1].localMax > n) { // If the previous block has a higher local maximum**

**stack->S[stack->top].value = n; // Set the value of the block**

**stack->S[stack->top].localMax = stack->S[stack->top - 1].localMax; // Set the local maximum same as the previous block**

**}**

**else {**

**stack->S[stack->top].value = n; // Set the value of the block**

**stack->S[stack->top].localMax = n; // Set the local maximum as the value itself**

**}**

**}**

**printf("%d inserted in stack\n", n);**

**}**

**}**

**// Function to pop an element from the stack**

**void pop(Stack\* stack) {**

**if (stack->top == -1) { // Check if the stack is empty**

**printf("Stack is empty\n");**

**}**

**else {**

**stack->top--; // Decrement the top index**

**printf("Element popped\n");**

**}**

**}**

**// Function to find the maximum element in the stack**

**void max(Stack\* stack) {**

**if (stack->top == -1) { // Check if the stack is empty**

**printf("Stack is empty\n");**

**}**

**else {**

**printf("Maximum value in the stack: %d\n", stack->S[stack->top].localMax); // Print the local maximum of the top block**

**}**

**}**

**int main() {**

**Stack\* S1 = createStack(5); // Create a stack of size 5**

**push(S1, 2);**

**max(S1);**

**push(S1, 6);**

**max(S1);**

**pop(S1);**

**max(S1);**

**return 0;**

**}**