**Roll No…………….. Total No. of Pages:……**

**ST-1 (SET-III)**

**4th SEMESTER 2022-23**

**22CS006- Object Oriented Programming**

**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) Write 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) Create a student database using nested structures to store the name, roll number,and marks of three subjects for multiple students. Calculate and display the average marks for each student.

**Input**:

Enter name and roll number for student 1: Avinash 23

Enter marks for subject 1: 65

Enter marks for subject 2: 45

Enter marks for subject 3: 50

Enter name and roll number for student 2: Ravi 19

Enter marks for subject 1: 68

Enter marks for subject 2: 50

Enter marks for subject 3: 55

Enter name and roll number for student 3: Kavya 32

Enter marks for subject 1: 60

Enter marks for subject 2: 57

Enter marks for subject 3: 59

**Output**:

Student Database:

Name: Avinash, Roll Number: 23, Average Marks: 53.33

Name: Ravi, Roll Number: 19, Average Marks: 57.67

Name: Kavya, Roll Number: 32, Average Marks: 58.67

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) Write a C program to create Binary Search Tree using structure and perform inorder traversal on BST(Binary Search Tree).

Solution :

**#include <stdio.h>**

**#include <stdlib.h>**

**// Structure to represent a node in the binary search tree**

**struct Node {**

**int data;**

**struct Node\* left;**

**struct Node\* right;**

**};**

**// Function to create a new node with the given data**

**struct Node\* createNode(int data) {**

**// Allocate memory for the new node**

**struct Node\* newNode = (struct Node\*)malloc(sizeof(struct Node));**

**// Set the data and pointers of the new node**

**newNode->data = data;**

**newNode->left = NULL;**

**newNode->right = NULL;**

**return newNode;**

**}**

**// Function to insert a new node with the given data into the binary search tree**

**struct Node\* insert(struct Node\* root, int data) {**

**// If the tree is empty, create a new node and return it as the root**

**if (root == NULL) {**

**return createNode(data);**

**} else {**

**// If the data is less than the root's data, insert it into the left subtree**

**if (data < root->data) {**

**root->left = insert(root->left, data);**

**}**

**// If the data is greater than or equal to the root's data, insert it into the right subtree**

**else {**

**root->right = insert(root->right, data);**

**}**

**}**

**return root;**

**}**

**// Function to perform an inorder traversal of the binary search tree**

**void inorderTraversal(struct Node\* root) {**

**if (root != NULL) {**

**// Traverse the left subtree**

**inorderTraversal(root->left);**

**// Print the data of the current node**

**printf("%d ", root->data);**

**// Traverse the right subtree**

**inorderTraversal(root->right);**

**}**

**}**

**int main() {**

**struct Node\* root = NULL;**

**// Insert elements into the binary search tree**

**root = insert(root, 50);**

**insert(root, 30);**

**insert(root, 20);**

**insert(root, 40);**

**insert(root, 70);**

**insert(root, 60);**

**insert(root, 80);**

**// Perform inorder traversal and display the elements of the tree**

**printf("Inorder traversal: ");**

**inorderTraversal(root);**

**return 0;**

**}**