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

**ST-1 (SET-VI)**

**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. Which of the following is NOT a characteristic of a stack?
   1. LIFO (Last-In-First-Out) structure
   2. **Allows random access to elements**
   3. Uses two main operations: push and pop
   4. Operates on a restricted portion of memory
2. What is the return type of the fgetc() function?
   1. char
   2. **int**
   3. FILE \*
   4. void
3. Which of the following is NOT a possible application of a stack data structure?
   1. Function call stack
   2. Expression evaluation
   3. Undo/Redo operations
   4. **Graph traversal**
4. Which of the following is an application of queues?
   1. Depth-first search (DFS)
   2. **Breadth-first search (BFS)**
   3. Binary search
   4. Quick sort
5. What is the correct way to declare a pointer to a pointer in C++?
   1. **int\*\* ptr;**
   2. int\* ptr\*;
   3. int\*\* \*ptr;
   4. int\*\* \*ptr\*;
6. What is the process of a recursive function calling itself multiple times?
   1. Iteration
   2. Accumulation
   3. **Stacking**
   4. Unwinding
7. Which of the following is an example of function overloading?
   1. void display(int x)
   2. void display(float y)
   3. void display(int x, float y)
   4. **All of the above**
8. Which preprocessor directive is used to check if a macro is defined in C++?
   1. #include
   2. #define
   3. **#ifdef**
   4. #ifndef
9. Which of the following is true about inline function calls?
   1. **They have a higher execution speed than regular function calls.**
   2. They always result in smaller code size.
   3. They can be used to call functions from external libraries.
   4. They are resolved at runtime.
10. What is the size of a pointer variable in C++ on a 32-bit system?
    1. 2 bytes
    2. **4 bytes**
    3. 8 bytes
    4. Depends on the type of the pointer

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

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

11) How do you delete memory allocated to a pointer to a pointer?

a) delete ptr;

**b) delete[] ptr;**

c) delete \*ptr;

d) delete[] \*ptr;

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

int x = 5;

if (x < 10)

if (x > 2)

cout << "A";

else

cout << "B";

**a) A**

b) B

c) A followed by B

d) No output

13) What will be the output of the following C++ program?

#include <iostream>

#include <string>

#include <cstring>

using namespace std;

int main(int argc, char const \*argv[])

{

const char \*a = "Hello\0World";

cout<<a;

return 0;

}

**a) Hello**

b) World

c) Error

d) Hello World

14) What will be the output of the following C++ code?

#include <iostream>

using namespace std;

int main ()

{

int a, b, c;

a = 2;

b = 7;

c = (a > b) ? a : b;

cout << c;

return 0;

}

a) 12

b) 14

c) 6

**d) 7**

15) What will be the output of the following C++ code snippet?

#include <stdio.h>

#include<iostream>

using namespace std;

int main ()

{

int array[] = {0, 2, 4, 6, 7, 5, 3};

int n, result = 0;

for (n = 0; n < 8; n++)

{

result += array[n];

}

cout << result;

return 0;

}

a) 21

**b) 27**

c) 26

d) 25

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

Q16) HR wants to accept the skills level within particular range from the interviewee. Write a C Program to identify valid range of the given number by defining upper and lower limits given by HR.

Check valid range using preprocessor directive.

Constants should defined in program

MAX\_VALUE=100

MIN\_VALUE=0

**Input:** 50

**Output:** Value is within the valid range.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Test Case 1** | **Test Case 2** | **Test Case 3** |
| **Input** | 150 | -10 | 45 |
| **Output** | Value exceeds maximum limit. | Value is below minimum limit. | Value is within the valid range. |

Solution :

**#include <stdio.h>**

**#define MAX\_VALUE 100**

**#define MIN\_VALUE 0**

**void check(int value){**

**// This is a conditional statement using preprocessor directives**

**#if (value > MAX\_VALUE)**

**printf("Value exceeds maximum limit.\n");**

**#elif (value < MIN\_VALUE)**

**printf("Value is below minimum limit.\n");**

**#else**

**printf("Value is within the valid range.\n");**

**#endif**

**}**

**int main() {**

**int value = 50; // variable initialization**

**check(value);**

**return 0;**

**}**

Q17) Kids are learning even and odd numbers. Help them in their homework to check if number is even or odd. Make use of macros.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Test Case 1** | **Test Case 2** | **Test Case 3** |
| **Input** | 150 | 89 | 0 |
| **Output** | 150 is even | 89 is odd | 0 is even |

Solution :

**#include <stdio.h>**

**// Macro to check if a number is even**

**#define IS\_EVEN(num) ((num % 2) == 0)**

**void check(int num){**

**if (IS\_EVEN(num)) { // Check if the number is even**

**printf("%d is even.", num);**

**} else {**

**printf("%d is odd.", num);**

**}**

**}**

**int main() {**

**int num = 0;**

**check(num);**

**return 0;**

**}**

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

Q18) Traveler wants to visit the cities in particular order. Cities are represented as a adjacency matrix where 1 represents the route between two cities. Help traveler to visit cities depth wise.

**Example:**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Test Case 1** | **Test Case 2** | **Test Case 3** |
| **Input** | 0 1 0 0 1  1 0 1 1 1  0 1 0 1 0  0 1 1 0 1  1 1 0 1 0  starting city: 0 | 0 1 1 1  1 0 1 0  1 1 0 0  1 0 0 0  starting city: 0 | 0 1 0 1  1 0 1 1  0 1 0 1  1 1 0 0  starting city: 2 |
| **Output** | DFS Traversal: 0 4 3 2 1 | DFS Traversal: 0 3 2 1 | DFS Traversal: 2 3 0 1 |

Solution :

**#include <stdio.h>**

**#include <stdbool.h>**

**#define MAX\_SIZE 100**

**int adjMatrix[MAX\_SIZE][MAX\_SIZE];**

**bool visited[MAX\_SIZE];**

**int stack[MAX\_SIZE];**

**int top = -1;**

**// Push element onto the stack**

**void push(int value) {**

**stack[++top] = value;**

**}**

**// Pop element from the stack**

**int pop() {**

**return stack[top--];**

**}**

**// Check if the stack is empty**

**bool isStackEmpty() {**

**return top == -1;**

**}**

**// Perform DFS traversal using a stack**

**void dfs(int startVertex, int numVertices) {**

**push(startVertex);**

**visited[startVertex] = true;**

**printf("DFS Traversal: ");**

**while (!isStackEmpty()) {**

**int currentVertex = pop();**

**printf("%d ", currentVertex);**

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

**if (adjMatrix[currentVertex][i] && !visited[i]) {**

**push(i);**

**visited[i] = true;**

**}**

**}**

**}**

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

**}**

**int main() {**

**int numVertices;**

**printf("Enter the number of vertices: ");**

**scanf("%d", &numVertices);**

**printf("Enter the adjacency matrix:\n");**

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

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

**scanf("%d", &adjMatrix[i][j]);**

**}**

**}**

**// Initialize visited array**

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

**visited[i] = false;**

**}**

**int startVertex;**

**printf("Enter the starting city: ");**

**scanf("%d", &startVertex);**

**dfs(startVertex, numVertices);**

**return 0;**

**}**