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

**ST-1 (SET-II)**

**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 function is used to open a file in C?
   1. **fopen()**
   2. open()
   3. read()
   4. close()
2. Which of the following is true about a constant pointer?
   1. **The address it points to cannot be changed**
   2. The value it points to cannot be changed
   3. Both the address and the value it points to cannot be changed
   4. The constant pointer cannot be assigned to another pointer
3. What is the time complexity of the push operation in a stack?
   1. **O(1)**
   2. O(log n)
   3. O(n)
   4. O(n^2)
4. What is the purpose of the "delete" operator in C++?
   1. **To free the memory allocated using "new"**
   2. To delete a pointer variable
   3. To remove a reference to an object
   4. To deallocate stack memory
5. How can you change the value of a variable through a pointer to a pointer?
   1. Using the & operator.
   2. **Using the \* operator.**
   3. Using the -> operator.
   4. Using the \*\* operator.
6. What is the term used to describe a recursive function that calls itself more than once?
   1. Multirecursion
   2. Binary recursion
   3. **Mutual recursion**
   4. Indirect recursion
7. What happens if two functions are overloaded solely based on their return types?
   1. **The compiler throws an error due to ambiguity**
   2. The first defined function is always chosen by the compiler
   3. The last defined function is always chosen by the compiler
   4. The compiler randomly selects one of the functions
8. Which preprocessor directive is used to issue a warning message during compilation in C++?
   1. #include
   2. #define
   3. #pragma
   4. **#warning**
9. Which of the following operations is used to check if a queue is empty?
   1. **isEmpty()**
   2. isFull()
   3. enqueue()
   4. dequeue()
10. When should an inline function be used?
    1. **When the function is called frequently**
    2. When the function has a large code body
    3. When the function is used in a recursive manner
    4. When the function has variable argument lists

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

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

1. Consider the following code snippet:

int x = 10;

int y = 20;

const int\* ptr = &x;

ptr = &y;

Is the above code snippet valid?

1. **Yes, the code is valid**
2. No, it violates const correctness
3. The code will produce a warning but will execute correctly
4. None of the above
5. What does the following code snippet do?

int\*\* ptr = new int\*[5];

1. Allocates memory for an integer pointer.
2. Allocates memory for a pointer to an integer.
3. Allocates memory for a 2D array of integers.
4. **Allocates memory for an array of integer pointers.**
5. What will be the output of the following code snippet?

int i = 5;

do

{

cout << i;

i--;

} while (i > 0);

1. 5432
2. 543
3. **54321**
4. 543210
5. What would be printed from the following C++ program?

#include <iostream>

using namespace std;

int main()

{

int a = 32, \*ptr = &a;

char ch = 'A', &cho = ch;

cho += a;

\*ptr += ch;

cout << a << ", " << ch << endl;

return 0;

}

1. 97, A
2. 128, A
3. 97, a
4. **129, a**
5. Find the output of the following program.

main(){

Float a = 5;

switch(a){

Case 5: cout <<”OOP”;

Default: cout <<”CPP”;

}

}

1. OOP
2. CPP
3. **Error**
4. OOPCPP

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

Q16) Mona has given a set of open and closed brackets of each type such as ( ),[ ], { } by his teacher

and he is asked to construct a valid set of brackets from the given set. A set is said to be valid if

1) Open brackets must be closed by the same type of closing brackets.

2) Open brackets must be closed in the correct order.

You have to determine that the set constructed by Mona is valid or not.

**Input :** ({})

**Output:** The expression is balanced

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Test Case 1** | **Test Case 2** | **Test Case 3** |
| **Input** | (({})} | [{()}] | [()] |
| **Output** | The expression is not balanced | The expression is balanced | The expression is balanced |

Solution :

**#include <stdio.h>**

**#include <stdbool.h>**

**#include <string.h>**

**#define MAX\_SIZE 100**

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

**int top = -1;**

**//push element to top of the stack**

**void push(char ch) {**

**if (top == MAX\_SIZE - 1) {**

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

**return;**

**}**

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

**}**

**//remove element from top of the stack**

**char pop() {**

**if (top == -1) {**

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

**return -1;**

**}**

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

**}**

**//function to check for expression is balanced or not**

**bool isBalanced(char\* expression) {**

**int len = strlen(expression);**

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

**// if expression contains ( or { or [ push it to stack**

**if (expression[i] == '(' || expression[i] == '{' || expression[i] == '[')**

**push(expression[i]);**

**else if (expression[i] == ')' || expression[i] == '}' || expression[i] == ']') {**

**if (top == -1)**

**return false;**

**char topChar = pop();**

**//check for right opening and closing parenthesis**

**if ((expression[i] == ')' && topChar != '(') ||**

**(expression[i] == '}' && topChar != '{') ||**

**(expression[i] == ']' && topChar != '['))**

**return false;**

**}**

**}**

**return top == -1;**

**}**

**int main() {**

**char expression[100];**

**printf("Enter an expression: ");**

**scanf("%[^\n]%\*c", expression);**

**if (isBalanced(expression))**

**printf("The expression is balanced\n");**

**else**

**printf("The expression is not balanced\n");**

**return 0;**

**}**

Q17) Write a macro function C to find the square of given number.

**Input :** 5

**Output:** The square of 5 is 25

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Test Case 1** | **Test Case 2** | **Test Case 3** |
| **Input** | 9 | 4 | -3 |
| **Output** | The square of 9 is 81 | The square of 4 is 16 | The square of -3 is 9 |

Solution :

**#include <stdio.h>**

**//macro to calculate square of given number.**

**#define SQUARE(x) ((x) \* (x))**

**int main() {**

**int num = -2;**

**int result = SQUARE(num); //store the result in variable**

**printf("The square of %d is %d.\n", num, result);**

**return 0;**

**}**

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

Q18) Implement matrix multiplication using structures.

Each matrix has dimensions and elements.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Test Case 1** | **Test Case 2** | **Test Case 3** |
| **Input** | 1 2  3 4  1 2  4 3 | 2 3  -1 5  1 -3  -2 -4 | -2 1  0 4  6 5  -7 1 |
| **Output** | 9 8  19 18 | -4 -18  -11 -17 | -19 -9  -28 4 |

Solution :

**#include <stdio.h>**

**// Structure definition for Matrix**

**struct Matrix {**

**int rows;**

**int columns;**

**int elements[10][10];**

**};**

**// Function to perform matrix multiplication**

**void multiplyMatrix(struct Matrix mat1, struct Matrix mat2, struct Matrix\* result) {**

**int i, j, k;**

**// Checking if matrices can be multiplied**

**if (mat1.columns != mat2.rows) {**

**printf("Error: Matrices cannot be multiplied.\n");**

**return;**

**}**

**// Performing matrix multiplication**

**for (i = 0; i < mat1.rows; i++) {**

**for (j = 0; j < mat2.columns; j++) {**

**result->elements[i][j] = 0;**

**for (k = 0; k < mat1.columns; k++) {**

**result->elements[i][j] += mat1.elements[i][k] \* mat2.elements[k][j];**

**}**

**}**

**}**

**result->rows = mat1.rows;**

**result->columns = mat2.columns;**

**}**

**// Function to display a matrix**

**void displayMatrix(struct Matrix mat) {**

**int i, j;**

**for (i = 0; i < mat.rows; i++) {**

**for (j = 0; j < mat.columns; j++) {**

**printf("%d ", mat.elements[i][j]);**

**}**

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

**}**

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

**}**

**int main() {**

**// Creating matrix structure variables**

**struct Matrix mat1, mat2, product;**

**// Inputting matrix dimensions and elements**

**printf("Enter dimensions of the first matrix:\n");**

**printf("Rows: ");**

**scanf("%d", &mat1.rows);**

**printf("Columns: ");**

**scanf("%d", &mat1.columns);**

**printf("Enter elements of the first matrix:\n");**

**for (int i = 0; i < mat1.rows; i++) {**

**for (int j = 0; j < mat1.columns; j++) {**

**scanf("%d", &mat1.elements[i][j]);**

**}**

**}**

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

**printf("Enter dimensions of the second matrix:\n");**

**printf("Rows: ");**

**scanf("%d", &mat2.rows);**

**printf("Columns: ");**

**scanf("%d", &mat2.columns);**

**printf("Enter elements of the second matrix:\n");**

**for (int i = 0; i < mat2.rows; i++) {**

**for (int j = 0; j < mat2.columns; j++) {**

**scanf("%d", &mat2.elements[i][j]);**

**}**

**}**

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

**// Multiplying matrices**

**multiplyMatrix(mat1, mat2, &product);**

**// Displaying matrices and the product**

**printf("Matrix 1:\n");**

**displayMatrix(mat1);**

**printf("Matrix 2:\n");**

**displayMatrix(mat2);**

**printf("Product of matrices:\n");**

**displayMatrix(product);**

**return 0;**

**}**