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

**ST-1 (SET-IV)**

**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 purpose of the feof() function?
   1. **Checks for end-of-file indicator**
   2. Reads a character from the file
   3. Closes the file
   4. Returns the current file position
2. Which function is used to deallocate dynamically allocated memory in CPP?
   1. **free()**
   2. delete()
   3. malloc()
   4. calloc()
3. Which stack operation retrieves the top element without removing it?
   1. push
   2. pop
   3. **peek**
   4. isEmpty
4. Which operator is used to access the value stored at a pointer to a pointer?
   1. **\***
   2. &
   3. ->
   4. ::
5. Which of the following must be done to avoid infinite recursion?
   1. Include the <iostream> library
   2. Use a for loop instead of recursion
   3. **Implement a base case**
   4. Declare all variables before the recursion
6. Can the return type of an overloaded function be different?
   1. Yes
   2. **No**
   3. Depends on the function signature
   4. Depends on the number of parameters
7. Which preprocessor directive is used to skip compilation of a block of code in C++?
   1. #include
   2. #define
   3. **#pragma**
   4. #error
8. What is the significance of assigning NULL to a pointer after freeing the memory?
   1. **It prevents the pointer from becoming a dangling pointer.**
   2. It indicates that the memory has been deallocated.
   3. It allows the pointer to be reused.
   4. It helps in avoiding memory leaks.
9. Can an inline function have recursion?
   1. Yes
   2. **No**
10. Which of the following is a disadvantage of using an array-based queue?
    1. **Limited capacity**
    2. Slower insertion and deletion operations
    3. Difficulty in implementing enqueue and dequeue operations
    4. Inefficient memory utilization

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

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

11) What does the "const" qualifier signify in the following function prototype?

void print(const int\* arr, int size);

**a) The function cannot modify the array elements**

b) The function cannot modify the pointer variable

c) The function cannot modify the array size

d) The function cannot modify any local variable

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

int num = 10;

int\* ptr = &num;

int\*\* ptr2 = &ptr;

cout << \*\*ptr2;

a) 0

**b) 10**

c) Error: invalid indirection

d) Error: incompatible types

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

int i = 0;

while (i < 3)

{

cout << i;

i++;

if (i == 2)

continue;

cout << "X";

}

a) 0

b) 01X

c) 012X

**d) 0X12X**

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

#include <iostream>

int main() {

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

int\* ptr = arr;

std::cout << \*(ptr + 2) << std::endl;

return 0;

}

a) 1

b) 2

**c) 3**

d) 4

15) What will be the output of the following code?

#include <iostream>

int main() {

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

std::cout << sizeof(arr) / sizeof(arr[0]);

return 0;

}

a) 1

**b) 5**

c) 10

d) Compilation error

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

Q16) While reading different files reader wants to know length of file as number of characters present in it. Write a program that reads a file and counts the number of characters present in it.

**Input:** A text file named "input.txt" with content.

**Output:** Total character count.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Test Case 1** | **Test Case 2** | **Test Case 3** |
| **Input** | File= learn c programming | File= The IT industry is one which is not limited to  software development alone. | File= hope is a thing with feather |
| **Output** | 19 | 74 | 28 |

Solution :

**#include <stdio.h>**

**int main() {**

**FILE \*file;**

**char ch;**

**int count = 0;**

**file = fopen("input.txt", "r"); //opening th file in read mode**

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

**printf("Error opening file.\n");**

**return 1;**

**}**

**while ((ch = fgetc(file)) != EOF) {**

**count++; //count characters in file while its not reach to end of file**

**}**

**fclose(file); //close file**

**printf("Total characters: %d\n", count);**

**return 0;**

**}**

Q17) Evaluate a given postfix expression using a stack data structure.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Test Case 1** | **Test Case 2** | **Test Case 3** |
| **Input** | 52+83-\*4/ | 456\*+ | 35- |
| **Output** | 8 | 34 | -2 |

Solution :

**#include <stdio.h>**

**#include <stdbool.h>**

**#include <ctype.h>**

**#include <string.h>**

**#define MAX\_SIZE 100**

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

**int top = -1;**

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

**void push(int value) {**

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

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

**return;**

**}**

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

**}**

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

**int pop() {**

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

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

**return -1;**

**}**

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

**}**

**int evaluatePostfix(char\* expression) {**

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

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

**if (isdigit(expression[i]))**

**//push digits on expression to the stack**

**push(expression[i] - '0');**

**else {**

**//if operator is found then pop two topmost operands from stack**

**int operand2 = pop();**

**int operand1 = pop();**

**//perform operation on two operands based on operator**

**switch (expression[i]) {**

**case '+':**

**push(operand1 + operand2);**

**break;**

**case '-':**

**push(operand1 - operand2);**

**break;**

**case '\*':**

**push(operand1 \* operand2);**

**break;**

**case '/':**

**push(operand1 / operand2);**

**break;**

**case '%':**

**push(operand1 % operand2);**

**break;**

**}**

**}**

**}**

**return pop();**

**}**

**int main() {**

**char expression[100];**

**printf("Enter a postfix expression: ");**

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

**int result = evaluatePostfix(expression);**

**printf("Result: %d\n", result);**

**return 0;**

**}**

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

Q18) Write a C program that finds the first negative integer in every window of size 'k' in a given array.

The program should implement a queue data structure using an array and provide functions to add and remove elements from the queue.

The first negative integer in a window is the negative integer that appears first when moving the window through the array.

**Input**: { 12, -1, -7, 8, -15, 30, 16, 28 }

k=3

**Output**: -1 -1 -7 -15 -15 0

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Test Case 1** | **Test Case 2** | **Test Case 3** |
| **Input** | Array: -5 -1 7 8 15 -30 16  Window Size (k): 3 | 5 -1 -7 8 15  Window Size (k): 2 | Array: -5 -1 -7 8 -9  Window Size (k): 3 |
| **Output** | -5 -1 0 -30 -30 | -1 -1 -7 0 | -5 -1 -7 |

Solution :

**#include <stdio.h>**

**#include <stdlib.h>**

**#define SIZE 100**

**// Structure to represent a queue**

**struct Queue {**

**int arr[SIZE];**

**int front, rear;**

**};**

**// Function to create an empty queue**

**struct Queue\* createQueue() {**

**struct Queue\* queue = (struct Queue\*)malloc(sizeof(struct Queue));**

**queue->front = queue->rear = -1;**

**return queue;**

**}**

**// Function to check if the queue is empty**

**int isEmpty(struct Queue\* queue) {**

**return queue->front == -1;**

**}**

**// Function to add an element to the queue**

**void enqueue(struct Queue\* queue, int item) {**

**if (queue->rear == SIZE - 1) {**

**printf("Queue is full");**

**return;**

**}**

**queue->arr[++queue->rear] = item;**

**if (queue->front == -1) {**

**queue->front = 0;**

**}**

**}**

**// Function to remove an element from the queue**

**int dequeue(struct Queue\* queue) {**

**if (isEmpty(queue)) {**

**printf("Queue is empty");**

**return -1;**

**}**

**int item = queue->arr[queue->front];**

**if (queue->front == queue->rear) {**

**queue->front = queue->rear = -1;**

**} else {**

**queue->front++;**

**}**

**return item;**

**}**

**// Function to find the first negative integer in every window of size k**

**void findFirstNegativeInWindow(int arr[], int n, int k) {**

**struct Queue\* queue = createQueue();**

**// Traverse through each window of size k**

**for (int i = 0; i < n - k + 1; i++) {**

**// Enqueue all negative integers in the current window**

**for (int j = i; j < i + k; j++) {**

**if (arr[j] < 0) {**

**enqueue(queue, arr[j]);**

**break; // Only enqueue the first negative integer in the window**

**}**

**}**

**// If the queue is not empty, print the front element (first negative integer)**

**// Otherwise, print 0 to indicate no negative integer in the window**

**if (!isEmpty(queue)) {**

**printf("%d ", queue->arr[queue->front]);**

**} else {**

**printf("0 ");**

**}**

**// Dequeue elements that are no longer in the current window**

**while (!isEmpty(queue) && queue->front <= i) {**

**dequeue(queue);**

**}**

**}**

**}**

**int main() {**

**int arr[] = { -5, -1, -7, 8, -9};**

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

**int k = 3;**

**printf("Array: ");**

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

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

**}**

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

**printf("Window Size (k): %d\n", k);**

**printf("First Negative Integers in Windows: ");**

**findFirstNegativeInWindow(arr, n, k);**

**return 0;**

**}**