**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 read a character from a file?
   1. fscanf()
   2. **fgetc()**
   3. fgets()
   4. getchar()
2. What happens if you try to modify the value of a constant pointer?
   1. **It results in a compile-time error**
   2. It results in a runtime error
   3. The modification is allowed
   4. None of the above
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. Which of the following correctly describes the process of dynamically allocating memory in C++?
   1. New memory is created automatically when a pointer is declared
   2. Memory is allocated using the "malloc" function
   3. **Memory is allocated using the "new" operator**
   4. Memory is automatically deallocated when a pointer goes out of scope
5. What is the size of a pointer to a pointer in C++? (Consider 64-bit machine)
   1. It depends on the size of the data type being pointed to.
   2. 1 byte.
   3. 4 bytes.
   4. **8 bytes.**
6. What is tail recursion?
   1. A recursive function with multiple base cases
   2. A recursive function that does not use any base case
   3. **A recursive function where the recursive call is the last operation**
   4. A recursive function that reverses the order of operations
7. Which of the following is NOT a valid condition for function overloading in C++?
   1. Different number of parameters
   2. Different order of parameters
   3. Different return types
   4. **Different access specifiers**
8. Which preprocessor directive is used to define a constant value in C++?
   1. #include
   2. **#define**
   3. #ifdef
   4. #const
9. In a circular queue, if the rear pointer is at the end of the array, where does the next element get added?
   1. The beginning of the array
   2. The middle of the array
   3. **After the rear element, wrapping around to the front**
   4. It cannot be added since the queue is full
10. Which keyword can be used to force the compiler to ignore the "inline" keyword?
    1. extern
    2. static
    3. restrict
    4. **noinline**

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

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

1. Consider the following code snippet:

int x = 10;

const int\* ptr = &x;

What does the above code mean?

* 1. The pointer is constant and cannot be modified
  2. **The pointed value is constant and cannot be modified**
  3. Both the pointer and the pointed value are constant
  4. None of the above

1. What is the output of the following code snippet?

int\* ptr = new int(5);

cout << \*ptr;

delete ptr;

1. 0
2. 1
3. **5**
4. Garbage value
5. What is the output of the following code snippet?

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

int\* ptr = arr + 3;

cout << \*ptr;

1. 1
2. 2
3. 3
4. **4**
5. What is the output of the following recursive function?

int factorial(int n) {

if (n <= 1)

return 1;

else

return n \* factorial(n - 1);

}

int main() {

cout << factorial(5);

return 0;

}

1. 5
2. 15
3. **120**
4. 720
5. What will be the output of the following code snippet?

int x = 10;

if (x > 5)

{

cout << "Hello, ";

if (x < 15)

cout << "world!";

}

1. Hello
2. **Hello, world!**
3. world!
4. No output

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

Q16) You have given a string. Write a function that reverses a string using a stack data structure.

**Input :** hello

**Output:** olleh

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Test Case 1** | **Test Case 2** | **Test Case 3** |
| **Input** | JAVA | Stack Overflow | Pune |
| **Output** | AVAJ | wolfrevO kcatS | enuP |

Solution :

**#include <stdio.h>**

**#include <string.h>**

**#define MAX\_SIZE 100**

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

**int top = -1;**

**void push(char ch) {**

**//check for stack is full or not**

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

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

**return;**

**}**

**//push element to stack**

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

**}**

**char pop() {**

**//check for stack is empty or not**

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

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

**return -1;**

**}**

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

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

**}**

**void reverseString(char\* str) {**

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

**//push all characters of the string one by one**

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

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

**//pop all characters of the string one by one and store in same string**

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

**str[i] = pop();**

**//after this, str will contain reversed string**

**}**

**int main() {**

**char str[100];**

**printf("Enter a string: ");**

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

**printf("Original string: %s\n", str);**

**reverseString(str);**

**printf("Reversed string: %s\n", str);**

**return 0;**

**}**

Q17) You are tasked with developing a program that performs number swapping using macros.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Test Case 1** | **Test Case 2** | **Test Case 3** |
| **Input** | 5 10 | 18 93 | 9 7 |
| **Output** | Before swap: x = 5, y = 10  After swap: x = 10, y = 5 | Before swap: x = 18, y = 93  After swap: x = 93, y = 18 | Before swap: x = 9, y = 7  After swap: x = 7, y = 9 |

Solution :

**#include <stdio.h>**

**#define SWAP(a, b) do { \**

**int temp = (a); \**

**(a) = (b); \**

**(b) = temp; \**

**} while (0)**

**int main() {**

**int x = 5, y = 10;**

**printf("Before swap: x = %d, y = %d\n", x, y);**

**SWAP(x, y); //cal SWAP Macro**

**printf("After swap: x = %d, y = %d\n", x, y);**

**return 0;**

**}**

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

Q18) You are given an array of integers nums, there is a sliding window of size k which is moving from the very left of

the array to the very right. You can only see the k numbers in the window. Each time the sliding window moves right by one position.

Return the max sliding window.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Test Case 1** | **Test Case 2** | **Test Case 3** |
| **Input** | [1,3,-1,-3,5,3,6,7], k = 3 | [2, 3, 0, -6, 4, 3], k = 3 | [2, 3, 0, -6, 4, 3], k = 2 |
| **Output** | [3,3,5,5,6,7] | [3,3,4,4] | [3,3,0,4,4] |

Solution :

**#include<stdio.h>**

**#include<stdlib.h>**

**#define MAX\_SIZE 100**

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

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

**struct queue**

**{**

**int size;**

**int f;**

**int r;**

**int\* arr;**

**};**

**//check if queue is empty**

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

**if(q->r==q->f){**

**return 1;**

**}**

**return 0;**

**}**

**//check if queue is full**

**int isFull(struct queue \*q){**

**if(q->r==q->size-1){**

**return 1;**

**}**

**return 0;**

**}**

**//add element in queue**

**void enqueue(struct queue \*q, int val){**

**if(isFull(q)){**

**printf("This Queue is full\n");**

**}**

**else{**

**q->r++;**

**q->arr[q->r] = val;**

**// printf("Enqued element: %d\n", val);**

**}**

**}**

**//remove element from queue**

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

**int a = -1;**

**if(isEmpty(q)){**

**printf("This Queue is empty\n");**

**}**

**else{**

**q->f++;**

**a = q->arr[q->f];**

**}**

**return a;**

**}**

**void BFS(int start,int numVertices){**

**// Initializing Queue (Array Implementation)**

**struct queue q;**

**q.size = 400;**

**q.f = q.r = 0;**

**q.arr = (int\*) malloc(q.size\*sizeof(int));**

**// BFS Implementation**

**int node;**

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

**visited[start] = 1;**

**enqueue(&q, start); // Enqueue i for exploration**

**while (!isEmpty(&q))**

**{**

**int node = dequeue(&q);**

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

**{**

**if(adjMatrix[node][j] ==1 && visited[j] == 0){**

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

**visited[j] = 1;**

**enqueue(&q, j);**

**}**

**}**

**}**

**}**

**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] = 0;**

**}**

**int startVertex;**

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

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

**BFS(startVertex, numVertices);**

**return 0;**

**}**