



**INSTRUCTIONS TO CANDIDATES:**



1. What do you understand by the following terms in Data Structures: (**6 marks)**
   1. Array
   2. Linked list.
   3. Stack
   4. Queue
   5. Namespace
   6. Structure

**Sample Answers:**

**Array**

Collection of elements of the same datatype

**Linked list.**

A linked list is a **linear dynamic data structure**, in which the elements are not stored at contiguous memory locations. A linkedlist consists of links where each link has **data** and a link to the next element

**Stack**

A stack is a linear data structure that follows the principle of Last In First Out (LIFO). This means the last element inserted inside the stack is removed first..

**Queue**

A queue is a linear data structure that stores the elements sequentially. It uses the FIFO approach (First In First Out) for accessing elements.

One end (rear end or the tail) is always used to insert data (enqueue) and the other end(front end or the head) is used to remove data (dequeue).

**Namespace**

A namespace is a declarative region that provides a scope to the identifiers (the names of types, functions, variables, etc) inside it. They are used to organize code into logical groups and to prevent name collisions that can occur especially when your code base includes multiple libraries.

**Structure**

A structure is a user-defined data type in C/C++. A structure is a collection of variables of different data types under a single name. A structure creates a data type that can be used to group items of possibly different types into a single type.

1. How do you define a constant variable in C++? (**2 marks)**

#define PI 3.14

const int I = 5

1. Given an array A of **n** elements, explain the following bubble sort sorting algorithm complexities of **A**: (**3 marks)**
2. Best case Time Complexity
3. Worst-case time complexity
4. Average case complexity

**Answers:**

Best case Time Complexity: Array is sorted O(n)

Worst-case time complexity: Array is not sorted O(n2)

Average case complexity: Some elements are sorted O(n2)

1. Given a singly linked list of numbers identified by a globally accessible head node“**head” and the node class defined below** ,develop a function **deleteNode** that would Delete the first occurring node whose data section contains k as its value.

The function signature is **void deleteNode(int k);**

// The node

**class** **Node** {  
 **int** data;  
 **Node**\* **next**;  
   
 };

void deleteNode(int k) {

Node \*current = head;

Node \*prev = NULL’

while (current != NULL && current->data != k) {

prev = current;

current = current->next;

}

if (current != NULL) {

if (prev == NULL) {

head = current->next;

} else {

prev->next = current->next;

}

delete current;

} else {

cout<<” Node not found”<<endl;

}

}

(**4 marks)**

1. Conclude on the output of the following c++ program: (**2 marks)**

///////////////////////////

#include <cstdlib>

#include <iostream>

using namespace std;

int main() {

short x=0b010100001;

short y=0b00110000;

short sum = x + y;

cout << "The sum of " << x << " and " << y <<" is :"<< sum << endl;

**return** 0;

}

**Answer:**

The sum of 161 and 48 is :209

1. Given an unsorted array **A** of **n** elements, write a complete C++ program that would demonstrate a bubble sort algorithm on **A**. (**3 marks)**

**Sample Answer**

#include<iostream>

using namespace std;

int main(){

int i,count,j,temp,n=8;

int A[8]={12,3,1,5,18,10,7,35};

cout<<"Unsorted array"<<endl;

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

cout<< A[i]<<"\t";

}

cout<<endl;

cout<<endl;

**for(count=1;count<=n-1;count++){**

**for(j=0;j<=n-2;j++)**

**{**

**if(A[j+1]<A[j]){**

**temp=A[j];**

**A[j]=A[j+1];**

**A[j+1]=temp;**

**}**

**}**

**}**

cout<<"Sorted array"<<endl;

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

cout<< A[i]<<"\t";

}

cout<<endl;

return 0;

}

1. Given a node **ntext\_node** of type **struct Node** as defined below, **next\_node** being part of a doubly linked list; write a function that would add a node before the **next\_node** node.

////////////////

// Definition of the node  
 **struct** **Node** {  
 **int** data;  
 **struct** **Node**\* **next**;  
 **struct** **Node**\* **prev**;  
 };

(**4 marks)**

**Sample ANswer**

void addNodeBefore(struct Node\* next\_node, int new\_data) {

if (next\_node == NULL) {

return;

}

Node \*new\_node = new Node(new\_data);

new\_node->prev = next\_node->prev;

new\_node->next = next\_node;

if (next\_node->prev != NULL) {

next\_node->prev->next = new\_node;

}

next\_node->prev = new\_node;

}

1. Give the time complexity of push, pop, enqueue, and dequeue in stack and queue data structures respectively. (**2 marks)**

**Answer:**

O(1) for push, pop, enqueue, and dequeue in stack and queue

1. Stack and queue are often classified as independent data structures rather than "limited access linked lists." Discuss any one reason behind this classification and how they differ from traditional linked lists. (**2 marks)**

LIFO(Stack) vs FIFO(Queue)

1. Give any two stack applications and queue any two que applications in the real world**.** (**2 marks)**

**Stack**

reverse a word

compiler

browsers

**Queue**

People on an escalator

Cashier line in a store

A car wash line

One way exits

**Section B: Answer ONLY ONE (1) question.** (**10 marks)**

1. Write a function named **countDigit** that returns the number of times that a given digit appears in a positive number. For example countDigit(32121, 1) would return 2 because there are two 1s in the number 32121. Other examples:

countDigit(33331, 3) returns 4

countDigit(33331, 6) returns 0

countDigit(3, 3) returns 1

int countDigit<(int numb, int n)

{

int counter=0;

string number=to\_string(numb);

if (numb<0){

return -1;}

else{

for(int i=0; i<number.length();i++){

int k=int(number[i])-48; //Remove the ASCII value of character 0

if(k==n){

counter++;

}

}

return counter;

}}

1. Write the pseudocode of algorithms to do the following: (**10 marks)**
2. PUSH and POP operations on a stack.
3. Dequeue and enqueue on Queue data structure
4. Display data in a singly linked list.
5. Insert a new node at the beginning of a singly linked list.
6. Using an example, explain how to implement the queue data structure using an array. (**10 marks**

#include <iostream>

using namespace std;

int queue[100], n = 100, front = - 1, rear = - 1;

void enqueue() {

int val;

if (rear == n - 1)

cout<<"Queue Overflow/Full"<<endl;

else {

if (front == - 1)

front = 0;

cout<<"Insert the element in queue : "<<endl;

cin>>val;

rear++;

queue[rear] = val;

}

}

void dequeue() {

if (front == - 1 || front > rear) {

cout<<"Queue Underflow/Empty ";

return ;

}

cout<<"Element deleted from queue is : "<< queue[front] <<endl;

// No need of shifting left , but this is not a better implementation

front++;

}

void display() {

if (front == - 1)

cout<<"Queue is empty"<<endl;

else {

cout<<"Queue elements are : ";

for (int i = front; i <= rear; i++)

cout<<queue[i]<<" ";

cout<<endl;

}

}

void readFront(){

if(front==-1){

cout<<"Queue underflow"<<endl;

return ;

}

cout<<"Element at front is:"<<queue[front]<<endl;

}

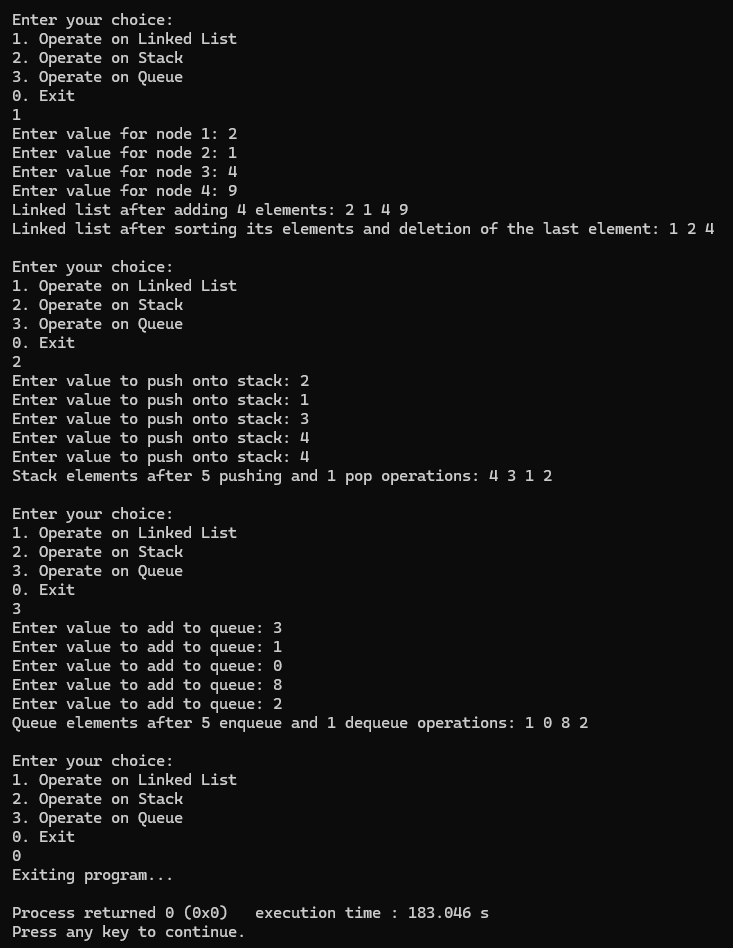
**Section C: Practice:** (**60 marks)**

1. Write a C++ program to implement the following data structures’ operation: linked list, stack and queue depending on the user's input number (1, 2 and 3 respectively). The program should continue to run and exit only if the user enters 0.

For every data structure, operations to be implemented are as follow:

* linked list: add 4 nodes, sort its elements, and delete the last element. (**30 marks)**
* stack: push 5 elements, pop one element (after checking if the stack is full or empty respectively). (**10 marks)**
* Queue: add five elements and remove/delete one element. (**10 marks)**

Refer to the following sample output:



Other requirements:

* + 1. Exit the program when the user’s choice is 0.  **(2 marks)**
    2. Code indentation **(2 marks)**
    3. Code comments **(2** **marks)**
    4. Use of functions **(2 marks)**
    5. Use of variables/identifier **(2 marks)**

**GOOD LUCK!**