

**CAT**

ACADEMIC YEAR: **2023-2024**

LEVEL: **Y2 (Level 4)**

TERM: **2**

COURSE TITLE: **DATA STRUCTURES AND ALGORITHMS WITH C++**

MODULE CODE: **SPEDA402**

NUMBER OF TEACHING HOURS/WEEK: **5**

DATE: **01/03/2024**

DURATION: **3 hours**

MAXIMUM MARKS: **50**

**INSTRUCTIONS:**

1. This question paper is made of 12 questions.
2. Try to answer **ALL QUESTIONS.**
3. Documents and computers are closed**.**
4. Write all answers on the provided answer sheet with a readable handwriting.

**Instructors:**

* **HABANABASHAKA Jean Damascene**
* **HATANGIMBABAZI Hilaire**
  1. Give the algorithms of PUSH and POP operations on a stack **(2 marks).**

***Answer:***

PUSH algorithm:

1. Check if the stack is full or not.
2. If the stack is full, then print an error of overflow and exit the program.
3. If the stack is not full, add the element and increment the top.

POP algorithm:

1. Check if the stack is empty or not.
2. If the stack is empty, then print an error of underflow and exit the program.
3. If the stack is not empty, then print the element at the top and decrement the top.
   1. Give any two applications of Stack data structure **(2 marks)**

**Answer:**

Stack applications:

* Browser history navigation
* Parenthesis checking
* Reverse a word.
* Calculate the value of expressions.
* ……..
  1. Explain the limitations of arrays in storing linear data. **(4 marks)**

**Answer:**

* **The size of the arrays is fixed**: So, we must know the upper limit on the number of elements in advance. Also, generally, the allocated memory is equal to the upper limit irrespective of the usage.
* **Insertion of a new element / Deletion of an existing element in an array of elements is expensive:** The room has to be created for the new elements and to create room existing elements have to be shifted but in Linked list if we have the head node then we can traverse to any node through it and insert new node at the required position.
  1. Give 2 advantages and 2 disadvantages of using linked lists. **(4 marks)**

**Answer:**

**Advantages:**

* Dynamic Array.
* Ease of Insertion/Deletion compared to contiguous (array) list.
* Overflow can never occur unless the memory is full.

**Disadvantages:**

* Pointers require extra space.
* Linked lists does not allow random access. We must access elements sequentially starting from the first node (head node). So, we cannot do a binary search with linked lists efficiently with its default implementation.
* Time must be spend traversing and changing the pointers.
* Programming is typically tricky with pointers.
* Reverse traversing is not possible in singly linked lists.
* Direct access to an element is not possible in a linked list as in an array by index.
* Searching for an element is costly and requires O(n) time complexity.
* Sorting of linked lists is very complex and costly.
  1. What do you understand by the following: **(3 marks)**

1. Singly linked list: A singly linked list is a data structure in which each element, known as a node, contains a value and a reference to the next node in the sequence. The last node typically points to null, indicating the end of the list.
2. Doubly linked list: A doubly linked list is a data structure like a singly linked list, but each node contains references to both the next and previous nodes in the sequence.
3. Circular linked list: A circular linked list is a variation of a linked list where the last node points back to the first node, forming a circular structure.
   1. Give the algorithms for: (**4 marks)**
4. Display data in a singly linked list.

**STEP 1:** SET PTR = HEAD

**STEP 2:** IF PTR = NULL

 WRITE "EMPTY LIST"  
 GOTO STEP 7  
 END OF IF

**STEP 4:** REPEAT STEP 5 AND 6 UNTIL PTR = NULL

**STEP 5:** PRINT PTR→ DATA

**STEP 6:** PTR = PTR → NEXT

[END OF LOOP]

**STEP 7:** EXIT

1. Insert a new node at the beginning of a singly linked list.

**STEP 1**: DECLARE A HEAD POINTER AND MAKE IT AS NULL.

**STEP 2:** CREATE A NEW NODE WITH THE GIVEN DATA.

**STEP 3:** MAKE THE NEW NODE POINTS TO THE HEAD NODE.

**STEP 4:** FINALLY, MAKE THE NEW NODE AS THE HEAD NODE.

1. Delete a node at the end of a singly linked list.

DELETE\_AT\_END(HEAD):

IF HEAD IS NULL OR HEAD.NEXT IS NULL:

RETURN NULL

CURRENT = HEAD

WHILE CURRENT.NEXT.NEXT IS NOT NULL:

CURRENT = CURRENT.NEXT

CURRENT.NEXT = NULL

RETURN HEAD

1. Insert a new node at the end of a singly linked list.

INSERT\_AT\_END(HEAD, DATA):

NEW\_NODE = NODE(DATA)

IF HEAD IS NULL:

HEAD = NEW\_NODE

RETURN HEAD

CURRENT = HEAD

WHILE CURRENT.NEXT IS NOT NULL:

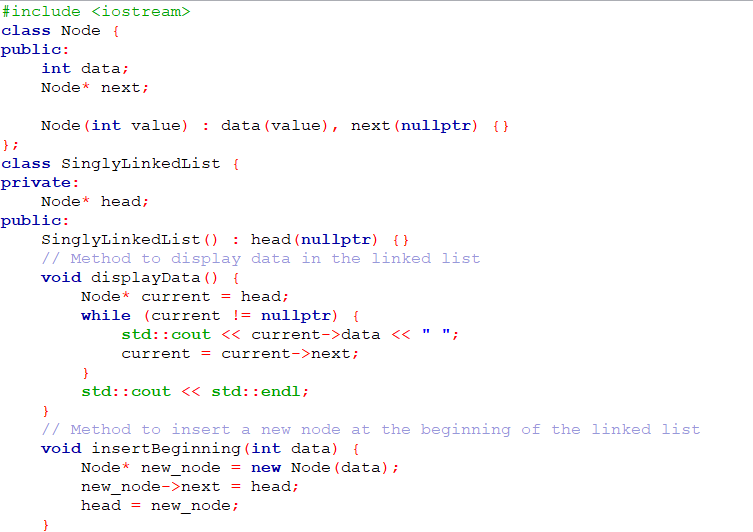
CURRENT = CURRENT.NEXT

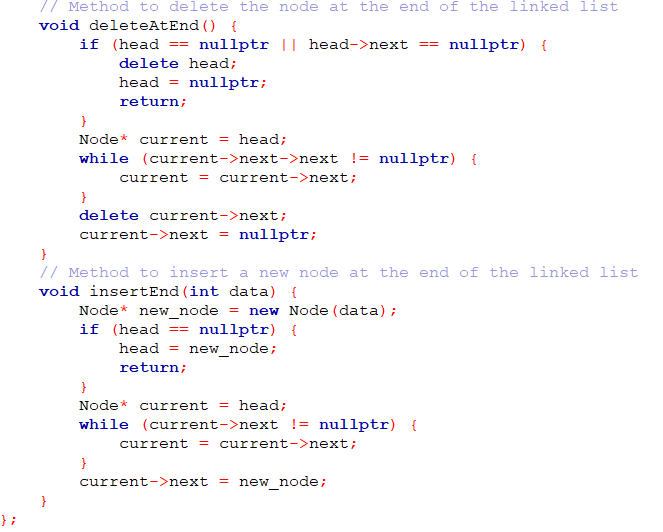
CURRENT.NEXT = NEW\_NODE

RETURN HEAD

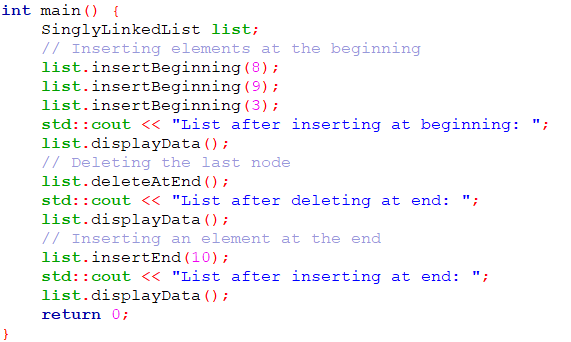
* 1. Using a C++ class called **Node**, implement the three algorithms in question 6, using methods/member functions ***displayData(), insertBeginning(),******deleteAtEnd()*** and ***insertEnd()*** respectively. **(8 marks)**

**//Some of possible programs**





//sample methods’ implementation



Corresponding output

A black background with white text

Description automatically generated

* 1. What are the time complexities of algorithms defined in questions 1 and 6? **4 marls**

**Answer:**

In question 1, the time complexity of both PUSH and POP algorithms is O (1).

Times complexities of algorithms in question 6:

* display data in a singly linked list: ***O(n)***
* Insert a new node at the beginning of a singly linked list: ***O (1)***
* Delete a node at the end of a singly linked list: The time complexity of an algorithm to delete a node at the end of a singly linked list is ***O (n)***, where n is the number of nodes in the linked list
* Insert a new node at the end of a singly linked list: ***O (1)*** if the tail node is maintained. If there is no direct reference to the tail node, traversing the entire list to find the last node will take linear time ***O*(*n*)**, where ***n*** is the number of nodes in the linked list.
  1. Using array implementation of the queue data structure, define a function that would perform the enqueue operation on any given queue: “Q” identified by rear: “R” and front: “F”. **(6 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;

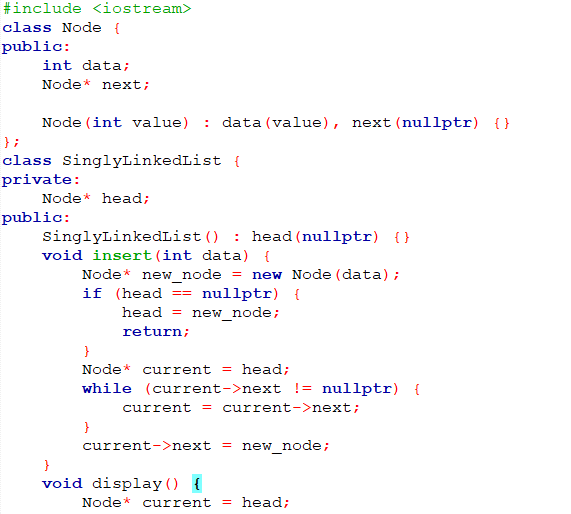
}

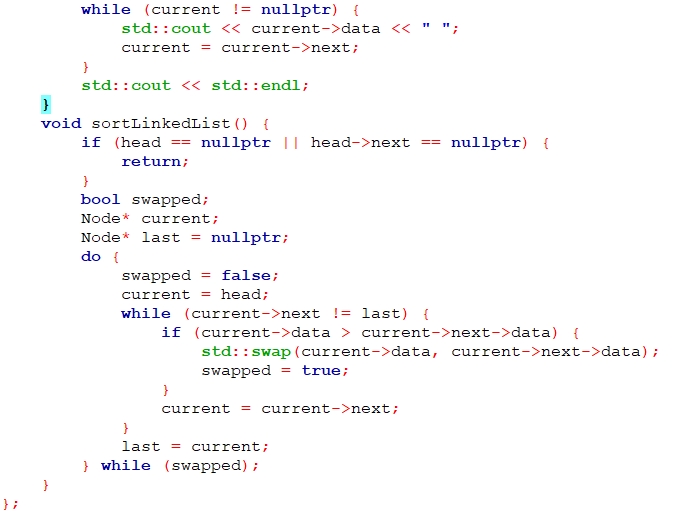
}

* 1. Write a C++ program implementing the sorting of a singly linked list (in ascending order). **(4 marks)**

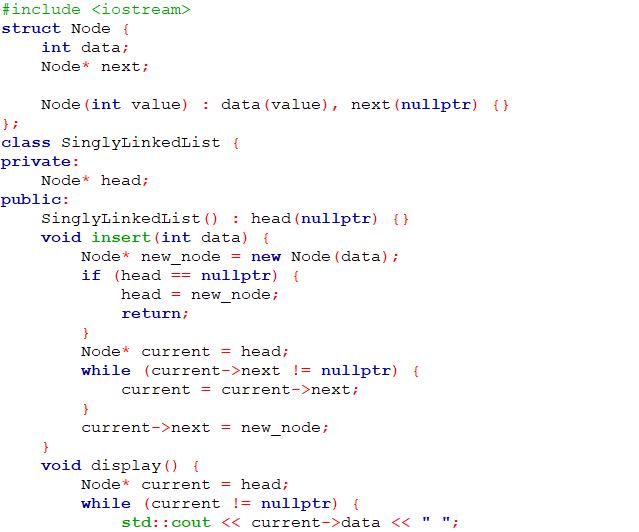
//Sample answers:

//using class





//Using struct



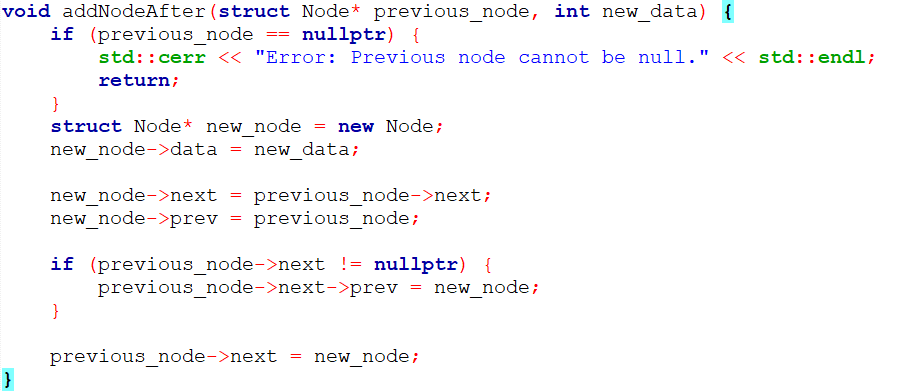
A screen shot of a computer code

Description automatically generated

* 1. Given a node previous\_node of type struct Node as defined below, previous\_node being part of a doubly linked list; write a function that would add a node after the previous\_node node. **(4marks)**

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

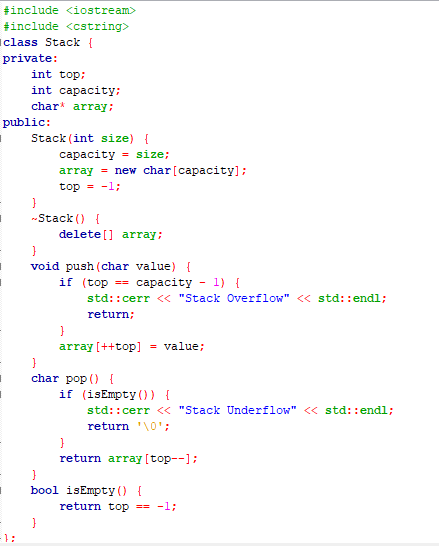
**Sample answer:**

****

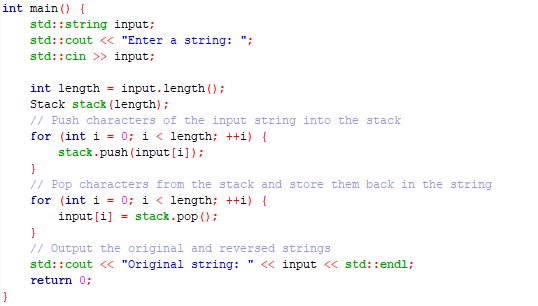
* 1. In a C++ program, implement the stack data structure using an array following the instruction given bellow:
* Create an empty stack.
* Push one by one all characters of any given string into the stack.
* Pop one by one all characters from the stack and store them in the same string.
* Output the string before and after popping.

What is your observation? **(5marks)**

**//Sample answer:**



//push() and pop() implementations in main() function



**Observation:** String reversing.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_GOOD LUCK \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**