AUA | American University of Afghanistan

Data Structures (ITC 310)

Major Assignment

Bushra Naeemi

ID: 29064

April 30, 2021

Q1. Create a circular link list with at least 7 nodes, now perform the following operation on that link list. As you perform the operation, write down the algorithm and c++ code too. Show the operations diagrammatically. (2 marks)

a. Insert a new node (insertion).
b. Delete from a link list.
c. Traverse

Answer:

a) Insertion: we will add two elements at the end of our circular linked list.

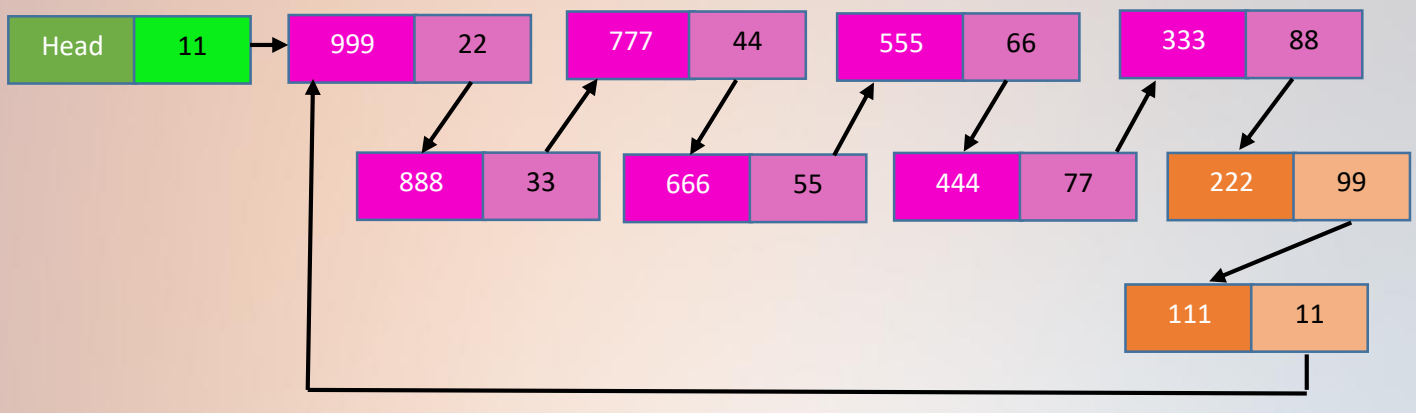
Insertion Algorithm:

```
temporary_node->insides = value;
temporary_node->next = NULL;

if(head == NULL){
    head = temporary_node;
}
else
{
while(second_temporary_node->next
!= NULL)
{
      second_temporary_node =
      second_temporary_node->next;
}
      second_temporary_node->next=
      temporary_node;
}
```

Circular Linked List Algorithm:

```
while(temporary_node->next != NULL)
        {
     temporary_node =
     temporary_node->next;
        }
        temporary_node->next =head;
```

--Diagram--

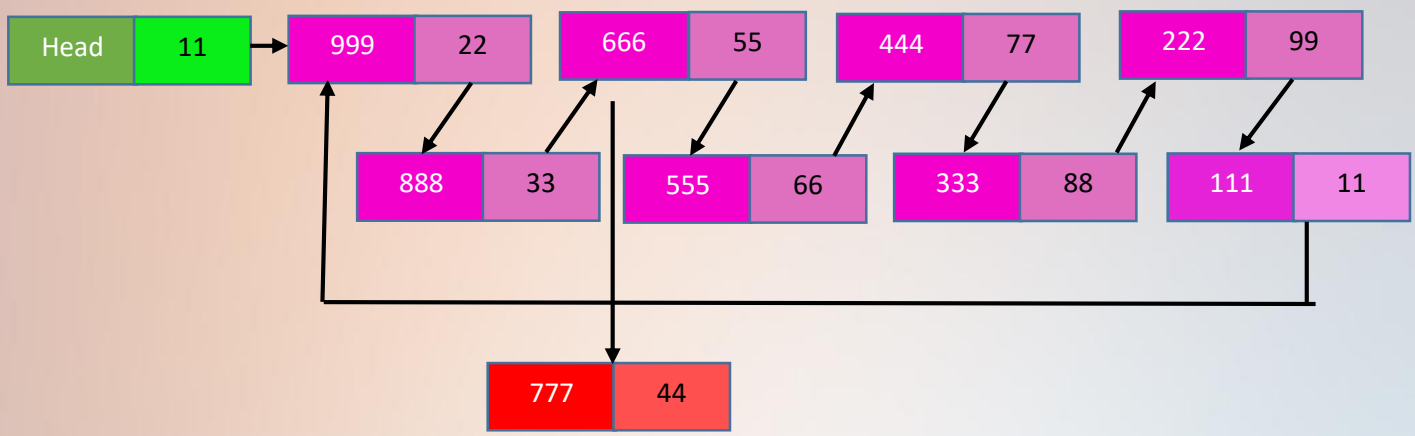


b) Deletion: Deleting a node from our circular linked list.

Algorithm:

```
bool warning = false;
while(temporary_node->next != head) {
      if(temporary_node->insides == input){
          warning = true;
          break;
     }
       last_one = temporary_node;
        temporary_node = temporary_node->next;
       }
       delete temporary_node;
       last_one->next = next;
       if(warning == false){
     cout<< "The value you are searching for does not exist.
     Sorry!!!";
     }
```

--Diagram—

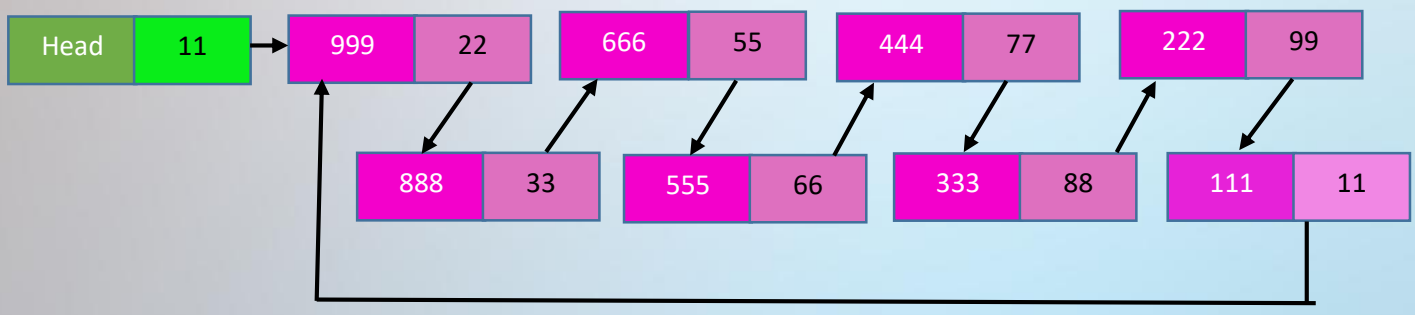


c) **Traversing:** Prinking the whole linked list.

Algorithm:

```
Cout<<temporary_node->insides;
temporary_node = temporary_node->next;
while(temporary_node != head)
        {
            cout << temporary_node->insides;
            temporary_node = temporary_node->next;
        }
```

--Diagram—



My C++ Codes for Q1:

C:\Users\Bushra Naeemi\Downloads\Q1.cpp - Dev-C++ 5.11

File Edit Search View Project Execute Tools AStyle Window Help

TDM-GCC 4.9.2 64-bit Release

(globals)

Pro Question1.cpp

```cpp
#include<iostream>
using namespace std;
struct noD
{
    int insides;
    struct noD* next;
};

struct noD* head=NULL;
//Main Class of making Circular Linked List

class circular_linkedList
{
public:
    //Inserstion method for section a) of Q1
    void node_insertion(int value)
    {
        struct noD* temporary_node = new noD();
        temporary_node->insides = value;
        temporary_node->next = NULL;
        if(head == NULL)
        {
            head = temporary_node;
        }
        else
        {
            struct noD* second_temporary_node = head;
            while(second_temporary_node->next != NULL)
            {
                second_temporary_node = second_temporary_node->next;
            }
            second_temporary_node->next = temporary_node;
        }
    }
```

```cpp
    //Making Circular Linked list method:
    void make_circular_linkedList()
    {
        struct noD* temporary_node = head;
        while(temporary_node->next != NULL)
        {
            temporary_node = temporary_node->next;
        }
        temporary_node->next = head;
    }
    //Delete method for section b) of Q1
    void node_deletion(int input)
    {
        struct noD* temporary_node = head;
        struct noD* last_one = temporary_node;
        bool warning = false;
        while(temporary_node->next != head)
        {
            if(temporary_node->insides == input){
                warning = true;
                break;
            }
            last_one = temporary_node;
            temporary_node = temporary_node->next;

        }
        struct noD* next=temporary_node->next;
        delete temporary_node;
        last_one->next = next;
        if(warning == false){
        cout<<endl<<"The value you are searching for does not exist. Sorry!!!"<<endl;
    }
    }
```

```cpp
    //Method for Making the Node Null:
    void making_node_null()
    {
        struct noD* temporary_node = head;
        while(temporary_node->next != head)
        {
            temporary_node = temporary_node->next;
        }
        temporary_node->next = NULL;
    }
    //Method For Traversing the list for c) of Q1:
    void traversing_nodes()
    {
        noD* temporary_node = head;
        cout << temporary_node->insides << " ";
        temporary_node = temporary_node->next;
        while(temporary_node != head)
        {
            cout << temporary_node->insides << " ";
            temporary_node = temporary_node->next;
        }
        cout<<endl;
    }
};
```

```cpp
//Main method to run the codes and test the results:
int main()
{
    circular_linkedList question_one;
    question_one.node_insertion(999);
    question_one.node_insertion(888);
    question_one.node_insertion(777);
    question_one.node_insertion(666);
    question_one.node_insertion(555);
    question_one.node_insertion(444);
    question_one.node_insertion(333);
    question_one.make_circular_linkedList();

    //Preview of inserted values in linked list:
    cout<<endl<<"Linked List Preview:"<<endl;
    question_one.traversing_nodes();

    //Insertion of 2 additional values at the end:
    cout<<endl<<"When I Insert The Values of 222 and 111"<<endl;
    question_one.making_node_null();
    question_one.node_insertion(222);
    question_one.node_insertion(111);
    question_one.make_circular_linkedList();
    question_one.traversing_nodes();

    //Deleting the value 777 from the list:
    cout<<endl<<"When I try to Delete 777:"<<endl;
    question_one.node_deletion(777);
    question_one.traversing_nodes();

    //Checking the method of deletion:
    cout<<endl<<"When I try to Delete Something that Doesn't exit in list: 1001"<<endl;
    question_one.node_deletion(1001);
    return 0;
}
```

Compiler | Resources | Compile Log | Debug | Find Results | Close

--The C++ codes is also in the folder if you want to run it--

--The Output--

```
C:\Users\Bushra Naeemi\Desktop\Bushra Major Assignment\Question1.exe

Linked List Preview:
999 888 777 666 555 444 333

When I Insert The Values of 222 and 111
999 888 777 666 555 444 333 222 111

When I try to Delete 777:
999 888 666 555 444 333 222 111

When I try to Delete Something that Doesn't exit in list: 1001

The value you are searching for does not exist. Sorry!!!

--------------------------------
Process exited after 0.05605 seconds with return value 0
Press any key to continue . . .
```

Q2. Create a doubly link list with at least 5 nodes, then perform the following operation on that link list. As you perform the operation, write down the algorithm and c++ code too. Show the operations diagrammatically. (3 marks).

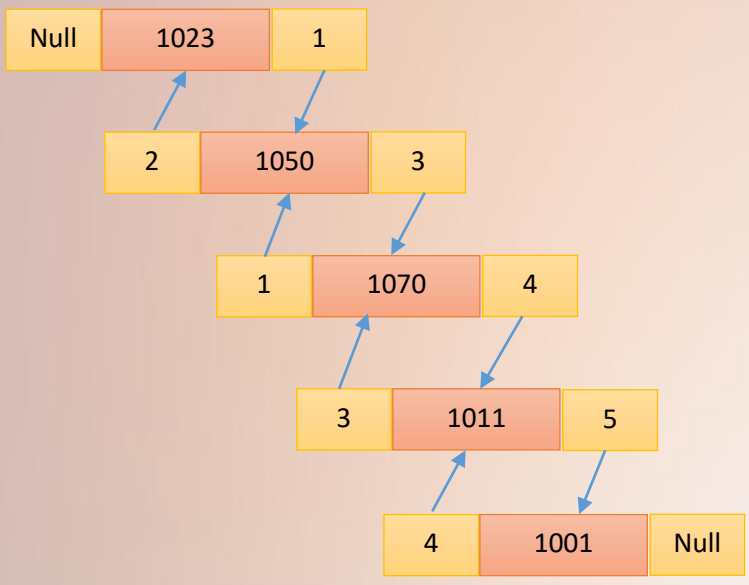
a. Traversal
b. Searching
c. Sorting

Answer:

a) Traversal. Algorithm
*In forward Direction:

```
noD* nodd = head;
noD* lastOne;
while (nodd != NULL){
       cout<<" "<<nodd->nodeData<<" ";
       lastOne = nodd;
       nodd = nodd->nextOne;
       }
```

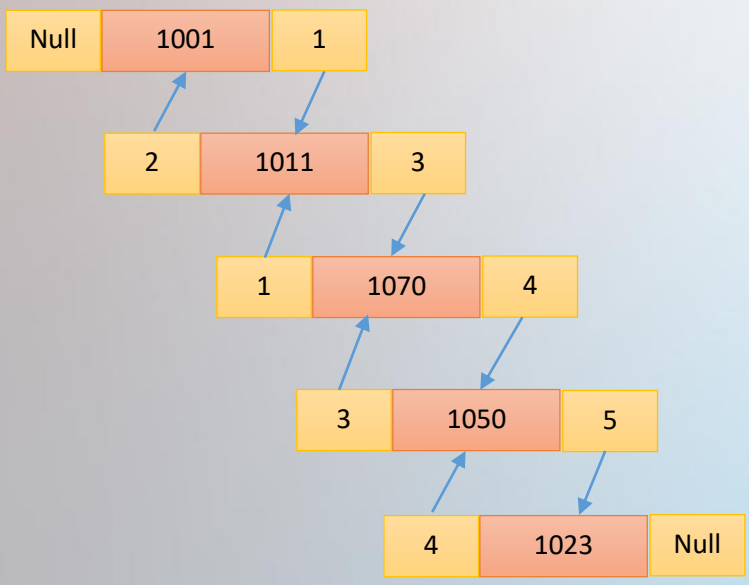
--Diagram—



*In reversed Direction:

```
while (lastOne != NULL){

        cout<<" "<<lastOne->nodeData<<" ";
        lastOne = lastOne->previousOne;}
```

--Diagram—

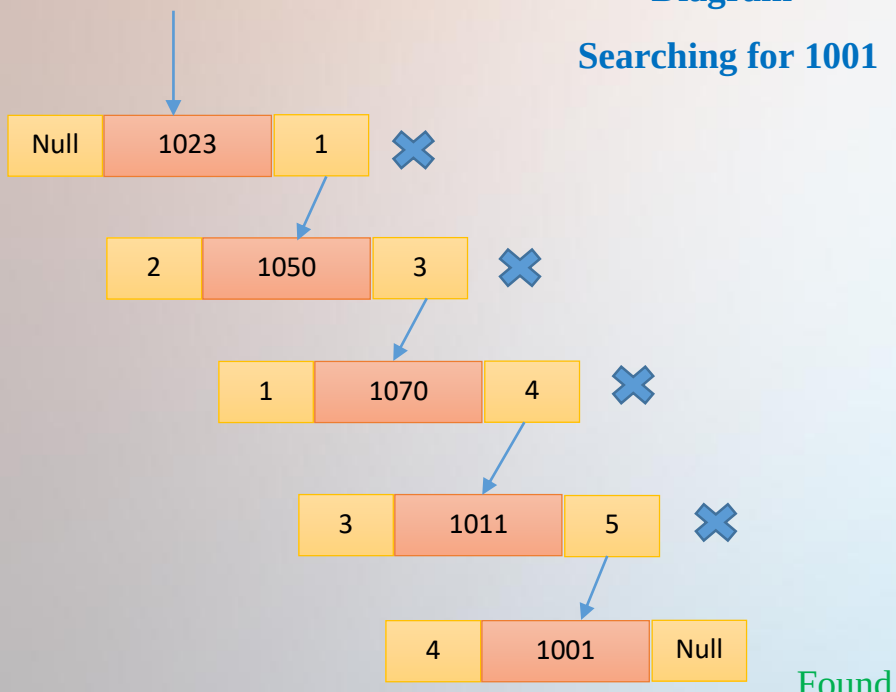


b) Searching Algorithm

```
noD* nodd = head;
int node_location=1;
bool not_there = false;
while (nodd != NULL){
        if(nodd->nodeData == inside_value){
                Print Location;
                not_there = true;
                break;
        }
        else if(nodd->nodeData != inside_value){
                nodd = nodd->nextOne;
                node_location++;
        }
        }
        if(not_there == false){

                cout<<endl<<"Intended values not Found."<<endl;
        }
```

--Diagram—

Searching for 1001



Found

c) Sorting

Algorithm:

```
if (head == NULL)
        return;
Do{
node_swap = 0;
node_pointerToNext = head;
        while (node_pointerToNext->nextOne != lastOne){
                if (node_pointerToNext->nodeData > node_pointerToNext->nextOne
                >nodeData){
                    swap(node_pointerToNext->nodeData, node_pointerToNext->nextOne-
                    >nodeData);
                    node_swap = 1;}
                 node_pointerToNext = node_pointerToNext->nextOne;}
                lastOne = node_pointerToNext;}
while (node_swap);
```

C++ Codes for Question 2:

Null	1023	1	2	1050	3	1	1070	4	3	1011	5	4	1001	Null

Pass 1

Null	1023	1	2	1050	3	1	1070	4	3	1011	5	4	1001	Null
Null	1023	1	2	1050	3	1	1070	4	3	1011	5	4	1001	Null
Null	1023	1	2	1050	3	1	1011	4	3	1070	5	4	1001	Null
Null	1023	1	2	1050	3	1	1011	4	3	1001	5	4	1070	Null

Pass 2

Null	1023	1	2	1050	3	1	1011	4	3	1001	5	4	1070	Null
Null	1023	1	2	1011	3	1	1050	4	3	1001	5	4	1070	Null

Null	1023	1	2	1011	3	1	1001	4	3	1050	5	4	1070	Null

Pass 3

Null	1011	1	2	1023	3	1	1001	4	3	1050	5	4	1070	Null

Null	1011	1	2	1001	3	1	1023	4	3	1050	5	4	1070	Null

Pass 4

Null	1001	1	2	1011	3	1	1023	4	3	1050	5	4	1070	Null

C++ Codes for Second Question:

\Bushra Naeemi\Desktop\Bushra Major Assignment\Question2.cpp - Dev-C++ 5.11

Search View Project Execute Tools AStyle Window Help

(globals)

Question2.cpp

```cpp
#include <bits/stdc++.h>
using namespace std;

class noD
{
    public:
    int nodeData;
    noD* nextOne;
    noD* previousOne;
};
noD* head = NULL;

//We First Need to Insert Some Values To Do The Operations
void node_insertion_at_end(int newOne)
{
    noD** headR = &head;
    noD* newNode = new noD();
    newNode->nodeData = newOne;

    noD* lastOne = *headR;
    newNode->nextOne = NULL;

    if (*headR == NULL)
    {
        newNode->previousOne = NULL;
        *headR = newNode;
        return;
    }

    while (lastOne->nextOne != NULL)
        lastOne = lastOne->nextOne;

    lastOne->nextOne = newNode;
    newNode->previousOne = lastOne;
}
```

```
//Method For Viewing or Printing All the List:
void nodes_traversing()
{
    noD* nodd = head;
    noD* lastOne;
    cout<<endl<<"Nodes Traversal in the Forward Direction Preview:"<<endl;

    while (nodd != NULL)
    {
        cout<<" "<<nodd->nodeData<<" ";
        lastOne = nodd;
        nodd = nodd->nextOne;
    }

    cout<<endl<<"Nodes Traversal in the Reversed Direction Preview:"<<endl;
    while (lastOne != NULL)
    {
        cout<<" "<<lastOne->nodeData<<" ";
        lastOne = lastOne->previousOne;
    }
}
//Method For Searching Nodes
void node_Searching(int inside_value)
{
    noD* nodd = head;
    int node_location=1;
    bool not_there = false;
    while (nodd != NULL)
    {
        if(nodd->nodeData == inside_value){

            cout<<endl<<"Your inserted value is in element location: "<<node_location<<endl;
            not_there = true;
            break;
        }
        else if(nodd->nodeData != inside_value){
            nodd = nodd->nextOne;
            node_location++;
        }

    }
    if(not_there == false){

        cout<<endl<<"Intended values not Found."<<endl;
```

```
    }

}
//Method For Sorting
void nodes_bubble_sort_method()
{
    int node_swap, x;
    noD* node_pointerToNext;
    noD* lastOne = NULL;

    if (head == NULL)
        return;
    do
    {
        node_swap = 0;
        node_pointerToNext = head;
        while (node_pointerToNext->nextOne != lastOne)
        {
            if (node_pointerToNext->nodeData > node_pointerToNext->nextOne->nodeData)
            {
                swap(node_pointerToNext->nodeData, node_pointerToNext->nextOne->nodeData);
                node_swap = 1;
            }
            node_pointerToNext = node_pointerToNext->nextOne;
        }
        lastOne = node_pointerToNext;
    }
    while (node_swap);
}
```

```
//main method to input values and text all the methods:

int main()
{
    //Value Insertion:
    node_insertion_at_end(1023);
    node_insertion_at_end(1050);
    node_insertion_at_end(1070);
    node_insertion_at_end(1011);
    node_insertion_at_end(1001);

    nodes_traversing();
    cout<<endl<<endl<<"---------------After Sorting: -----------------"<<endl<<endl;
    //Sorts the values then print:
    nodes_bubble_sort_method();
    nodes_traversing();
    //Searching For both existing and non existing values:
    cout<<endl<<endl<<"---------------Searching Values: -----------------"<<endl<<endl;
    node_Searching(1001);
    node_Searching(1000);
}
```

--You Can Also Run the Codes by Using the .cpp Folder in The File—

The Output:

```
C:\Users\Bushra Naeemi\Desktop\Bushra Major Assignment\Question2.exe

Nodes Traversal in the Forward Direction Preview:
 1023  1050  1070  1011  1001
Nodes Traversal in the Reversed Direction Preview:
 1001  1011  1070  1050  1023

---------------After Sorting: -----------------


Nodes Traversal in the Forward Direction Preview:
 1001  1011  1023  1050  1070
Nodes Traversal in the Reversed Direction Preview:
 1070  1050  1023  1011  1001

---------------Searching Values: -----------------


Your inserted value is in element location: 1

Intended values not Found.

--------------------------------
Process exited after 0.06346 seconds with return value 0
Press any key to continue . . .
```

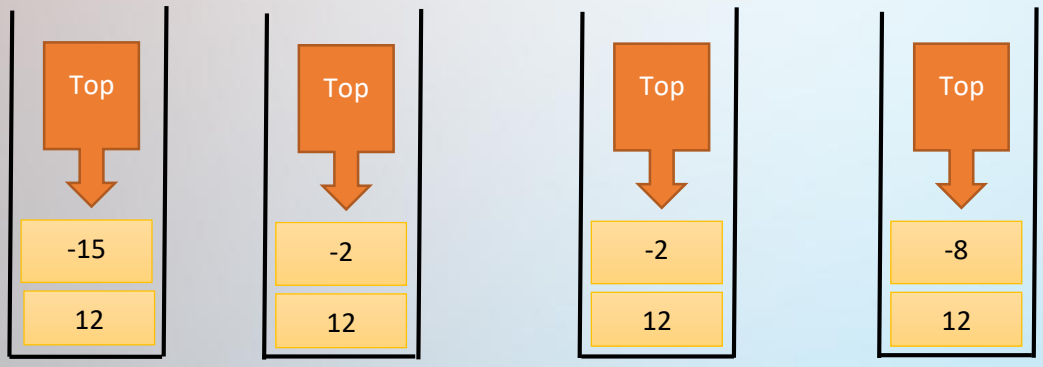
Q3. Consider the following operation on a stack. (2 marks).

```
S1.push (12);
S1.push (-15);
S1.pop();
S1.push (-2);
S1.push (-234);
Int t1 = S1.pop();
Int t3 = S1.pop();
S1.push (-8);
S1.push (20);
Int t2 = S1.topValue();
S1.pop ();
S1.push (4);
```

a. Assuming all instructions execute in the given sequence, draw four diagrams, showing the contents of the stack, after executing the second, fourth, sixth, and eighth instructions. In each diagram, include the values of all elements in the stack, and a pointer denoting the current "top" of the stack.

Answer:

Second Fourth Sixth Eight



b. What are the values of t1, t2, and t3 after the code executes?

Answer: Value of t1 is (-2). Value of t2 is (20). And value of t3 is (12).

c. Finally, give the C++ code that would implement the same sequence of additions and modifications to a stack. Note: implement the stack using Link List.

\Bushra Naeemi\Desktop\Bushra Major Assignment\Question3.cpp - Dev-C++ 5.11

Search View Project Execute Tools AStyle Window Help

(globals)

Question2.cpp | Question1.cpp | Question5.cpp | Question3.cpp

```cpp
#include <bits/stdc++.h>
using namespace std;
struct noD
{
    int insides;
    struct noD* node_address;
};
struct noD* tOP;
void push_Method(int insides)
{
    struct noD* temporary_node;
    temporary_node = new noD();

    if (!temporary_node)
    {
        cout << "------Stack Overflow-----";
        exit(1);
    }
    temporary_node->insides = insides;
    temporary_node->node_address = tOP;
    tOP = temporary_node;
    cout<<"push ("<<temporary_node->insides<<")"<<endl;
}
int pop_method()
{
    struct noD* temporary_node;
    if (tOP == NULL)
    {
        cout << "--------Stack Empty---------" << endl;
        exit(1);
    }
    else
    {
        temporary_node = tOP;
        tOP = tOP->node_address;
        temporary_node->node_address = NULL;
        cout<<"pop ("<<temporary_node->insides<<")"<<endl;
        return tOP->insides;
    }
}
```

```
void display_Method()
{
    struct noD* temporary_node;
        temporary_node = tOP;
        while (temporary_node != NULL)
        {
            cout<< temporary_node->insides << " ";

            temporary_node = temporary_node->node_address;
        }
        cout<<endl; }
int Empty_Check()
{
    return tOP == NULL;
}
int Value_of_Top()
{
    if (!Empty_Check())
        return tOP->insides;
    else
        exit(1);
}
int main()
{
    push_Method(12);
    push_Method(-15);
    cout<<"After Second Step: ";
    display_Method();
    cout<<endl;

    pop_method();
    push_Method(-2);
    cout<<"After Fourth Step:  ";
    display_Method();
    cout<<endl;

    push_Method(-234);
    int t1 = pop_method();
    cout<<"After Sixth Step:  ";
    display_Method();
    cout<<endl;

    int t3 = pop_method();
    push_Method(-8);
```

```
    int t3 = pop_method();
    push_Method(-8);
    cout<<"After Eighth Step:  ";
    display_Method();
    cout<<endl;

    push_Method(20);
    int t2 = Value_of_Top();
    pop_method();
    push_Method(4);
    cout<<"Finally:  ";
    display_Method();
    cout<<endl;

    cout<<"t1 : "<<t1<<"  t2 : "<<t2<<"  t3 : "<<t3;

    return 0;
}
```

--You Can Also Run the Code by Using .cpp file in Folder—

Output:

```
C:\Users\Bushra Naeemi\Desktop\Bushra Major Assignment\Question3.exe
push (12)
push (-15)
After Second Step: -15 12

pop (-15)
push (-2)
After Fourth Step:  -2 12

push (-234)
pop (-234)
After Sixth Step:  -2 12

pop (-2)
push (-8)
After Eighth Step:  -8 12

push (20)
pop (20)
push (4)
Finally:  4 -8 12

t1 : -2  t2 : 20  t3 : 12
--------------------------------
Process exited after 0.0512 seconds with return value 0
Press any key to continue . . .
```

Q4. Consider the following operation on a Queue. (2 marks)

```
Q1.enqueue (9);
Q1.enqueue (2);
Q1.enqueue (-2);
Q1.enqueue (-82);
Int t1 = Q1.frontValue ();
Q1.enqueue (-6);
Int t3 = Q1.dequeue ();
Q1.enqueue (14);
Int t2 = Q1.dequeue ();
Q1.enqueue (3);
```

a. Assuming all instructions execute in the given sequence, draw four diagrams, showing the contents of the queue, after executing the second, fourth, sixth, and eighth instructions. In each diagram, include the values of all elements in the queue, and two pointers denoting the current "front" and "rear" of the queue.

Answer:

Result of Second Execution:

9	2								
Front	Rear								
0	1	2	3	4	5	6	7	8	9

Result of Fourth Execution:

9	2	-2	-82						
Front			Rear						
0	1	2	3	4	5	6	7	8	9

Result of Sixth Execution:

9	2	-2	-82	-6					
Front				Rear					
0	1	2	3	4	5	6	7	8	9

Result of Eighth Execution:

	2	-2	-82	-6	14				
	Front				Rear				
0	1	2	3	4	5	6	7	8	9

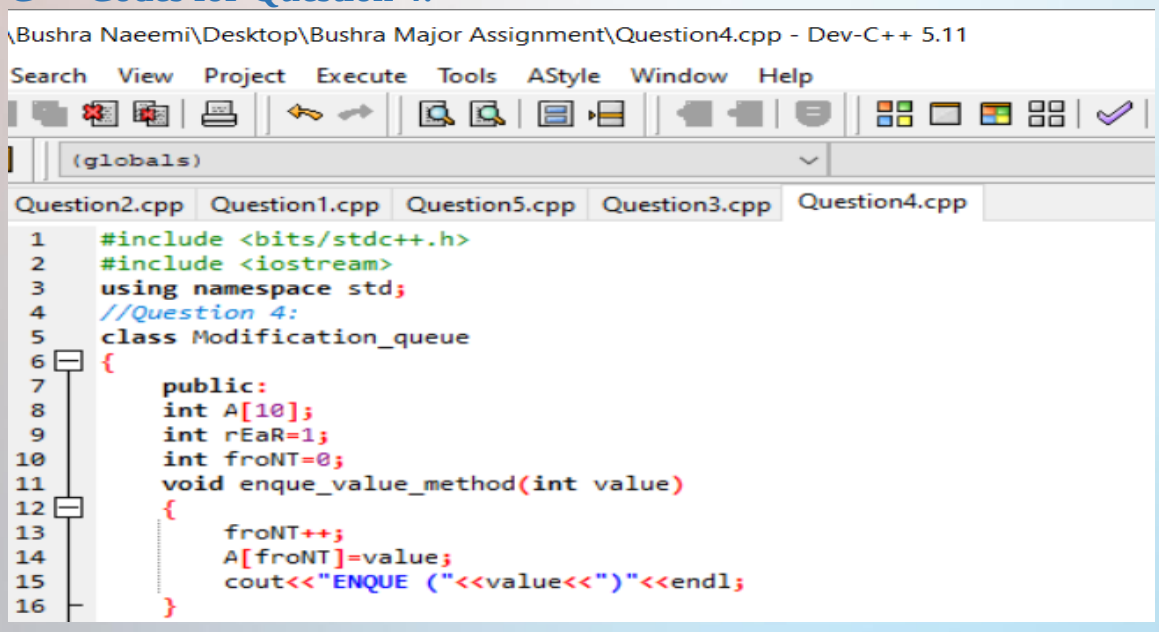
b. What are the values of t1, t2, and t3 after the code executes?

Answer:

- Value of t1 = -82
- Value of t2 = 2
- Value of t3 = 9

c. Finally, give the C++ code that would implement the same sequence of additions and modifications to a queue. Note: Implement the queue using array and not the link list.

Answer:
C++ Codes for Question 4:




```cpp
#include <bits/stdc++.h>
#include <iostream>
using namespace std;
//Question 4:
class Modification_queue
{
    public:
    int A[10];
    int rEaR=1;
    int froNT=0;
    void enque_value_method(int value)
    {
        froNT++;
        A[froNT]=value;
        cout<<"ENQUE ("<<value<<")"<<endl;
    }
```

```
    int deque_value_method()
    {
        int a = A[rEaR];
        if (froNT == 0) {
            cout<<endl<<"The Que is NULL"<<endl;
            return 0;
        }
        else {
            int value = A[rEaR];
            cout<<"DEQUE ("<<value<<")"<<endl;
            rEaR++;
        }
        return a; }
    void display_values_method()
    {
        int x;
        if (froNT == 0) {
            cout<<endl<<"The Que is NULL"<<endl;
            return;
        }

        for (x = rEaR; x <= froNT; x++) {
            cout<<A[x]<<"\t";
        }
        cout<<endl;
    }

```

```
    int froNTValue()
    {
        if (froNT == 0) {
            cout<<endl<<"The Que is NULL"<<endl;
            return 0;
        }

        return A[froNT];
    }
};
```

```
int main(void)
{

        Modification_queue Queue;
        Queue.enque_value_method(9);
        Queue.enque_value_method(2);
        cout<<"Step Two: "<<endl;
        Queue.display_values_method();
        cout<<endl;
        Queue.enque_value_method(-2);
        Queue.enque_value_method(-82);
        cout<<"Step Four: "<<endl;
        Queue.display_values_method();
        cout<<endl;
        int t1 = Queue.froNTValue();
        Queue.enque_value_method(-6);
        cout<<"Step Six: "<<endl;
        Queue.display_values_method();
        cout<<endl;
        int t3 = Queue.deque_value_method();
        Queue.enque_value_method(14);
        cout<<"Step Eight: "<<endl;
        Queue.display_values_method();
        cout<<endl;
        int t2 = Queue.deque_value_method();
        Queue.enque_value_method(3);
        Queue.display_values_method();

        Queue.deque_value_method();
        Queue.deque_value_method();
        Queue.display_values_method();

        cout<<"t1 : "<<t1<<"  t2 : "<<t2<<"  t3 : "<<t3;
    return 0;
}
```

--You Can Also Run the Code from Question4.ccp File—

The Output:

```
C:\Users\Bushra Naeemi\Desktop\Bushra Major Assignment\Question4.exe
ENQUE (9)
ENQUE (2)
Step Two:
9       2

ENQUE (-2)
ENQUE (-82)
Step Four:
9       2       -2      -82

ENQUE (-6)
Step Six:
9       2       -2      -82     -6

DEQUE (9)
ENQUE (14)
Step Eight:
2       -2      -82     -6      14

DEQUE (2)
ENQUE (3)
-2      -82     -6      14      3
DEQUE (-2)
DEQUE (-82)
-6      14      3
t1 : -82  t2 : 2  t3 : 9
--------------------------------
Process exited after 0.05334 seconds with return value 0
Press any key to continue . . .
```

Q5. Using a stack, you can "match" left and right parentheses by adding to the stack when you encounter a left parenthesis, and removing from the stack when you find a right parenthesis.

You should implement your solution as a function that takes a **string** input argument and returns a **bool** result. Assume that the string can be any series of characters -- such as an equation like **3 * (x + 4 / (9 - y))** -- and your method should ignore all characters expect the left, '(', and right, ')', parentheses.

Provide a main () function that includes a number of test cases. Make sure your code works for a variety of *positive examples* (those with an equal number of left and right parentheses) and *negative examples* (those with mismatching numbers of left and right parentheses -- be sure to test cases with both extra left parentheses and extra right parentheses). (2 marks).

Answer:

C++ Codes for Question 5: You can Also Run Question5.ccp in folder.

\Bushra Naeemi\Desktop\Bushra Major Assignment\Question5.cpp - Dev-C++ 5.11

Search View Project Execute Tools AStyle Window Help

TDM-GCC 4.9.2 64-bit Re

(globals)

Question2.cpp | Question1.cpp | Question5.cpp | Question3.cpp | Question4.cpp

```cpp
#include<iostream>
#include<stack>
#include<string>
using namespace std;
//Checks the for conditions
bool BracketPair(char start,char ending)
{
    if(start == '(' && ending == ')') return true;
    else if(start == '[' && ending == ']') return true;
    else if(start == '{' && ending == '}') return true;
    return false;
}
//Checks the balance of three types of brackets
bool EqualityCheck(string stringExpression)
{
    stack<char>  CharactersInString;
    for(int b =0;b<stringExpression.length();b++)
    {
        if(stringExpression[b] == '(' || stringExpression[b] == '{' || stringExpression[b] == '[')
            CharactersInString.push(stringExpression[b]);
        else if(stringExpression[b] == ')' || stringExpression[b] == '}' || stringExpression[b] == ']'){
            if(CharactersInString.empty() || !BracketPair(CharactersInString.top(),stringExpression[b]))
                return false;
            else
                CharactersInString.pop();
        }
    }
    return CharactersInString.empty() ? true:false;
}

int main()
{
    string yourExpression;
    cout<<"Input your string expression to do the Equality check for brackets: "<<endl;
    cout<<"==================================================================="<<endl;
    cin>>yourExpression;
    if(EqualityCheck(yourExpression))
        cout<<"All The Parenthesis In This String are Organized";
    else
        cout<<"The Parenthesis are Unorganized, You Need to Edit it! ";

};
```

Output of Positive Example: **3 * (x + 4 / (9 - y))**

C:\Users\Bushra Naeemi\Desktop\Bushra Major Assignment\Question5.exe

```
Input your string expression to do the Equality check for brackets:
===================================================================
3 * (x + 4 / (9 - y))
All The Parenthesis In This String are Organized
--------------------------------
Process exited after 3.126 seconds with return value 0
Press any key to continue . . .
```

Output of Positive Example: **{ 2 + 4 [5 * 4 (5 + 5)] }**

```
C:\Users\Bushra Naeemi\Desktop\Bushra Major Assignment\Question5.exe
Input your string expression to do the Equality check for brackets:
=====================================================================
{2+4[5*4(5+5)]}
All The Parenthesis In This String are Organized
--------------------------------
Process exited after 53.67 seconds with return value 0
Press any key to continue . . .
```

Output of Positive Example:

```
C:\Users\Bushra Naeemi\Desktop\Bushra Major Assignment\Question5.exe
Input your string expression to do the Equality check for brackets:
=====================================================================
7987(7987(87(98698)9879)987)
All The Parenthesis In This String are Organized
--------------------------------
Process exited after 16.06 seconds with return value 0
Press any key to continue . . .
```

Output of Negative Example (mismatching parenthesis):

```
C:\Users\Bushra Naeemi\Desktop\Bushra Major Assignment\Question5.exe
Input your string expression to do the Equality check for brackets:
=====================================================================
{()}[()]
The Parenthesis are Unorganized, You Need to Edit it!
--------------------------------
Process exited after 115 seconds with return value 0
Press any key to continue . . .
```

Output of Negative Example Extra Number of Parenthesis on the Left:

```
C:\Users\Bushra Naeemi\Desktop\Bushra Major Assignment\Question5.exe
Input your string expression to do the Equality check for brackets:
=====================================================================
{[((7980)+ (789)]
The Parenthesis are Unorganized, You Need to Edit it!
--------------------------------
Process exited after 137.1 seconds with return value 0
Press any key to continue . . .
```

Output of Negative Example Extra Number of Parenthesis on the Right:

```
C:\Users\Bushra Naeemi\Desktop\Bushra Major Assignment\Question5.exe
Input your string expression to do the Equality check for brackets:
=====================================================================
(700 + 687) + (799 * 789)]}
The Parenthesis are Unorganized, You Need to Edit it!
--------------------------------
Process exited after 74.32 seconds with return value 0
Press any key to continue . . .
```

Q6. For a directed graph, find if there is a path between two vertices or not. Write the algorithm and C++ code for it. Represent the example diagrammatically. Calculate the time and space complexity for it too. (3.5 marks).

Q6. 1: Record a video of a maximum 3minutes on Q6, and explain how it works. (1.5 marks).

Answer:

Algorithm:

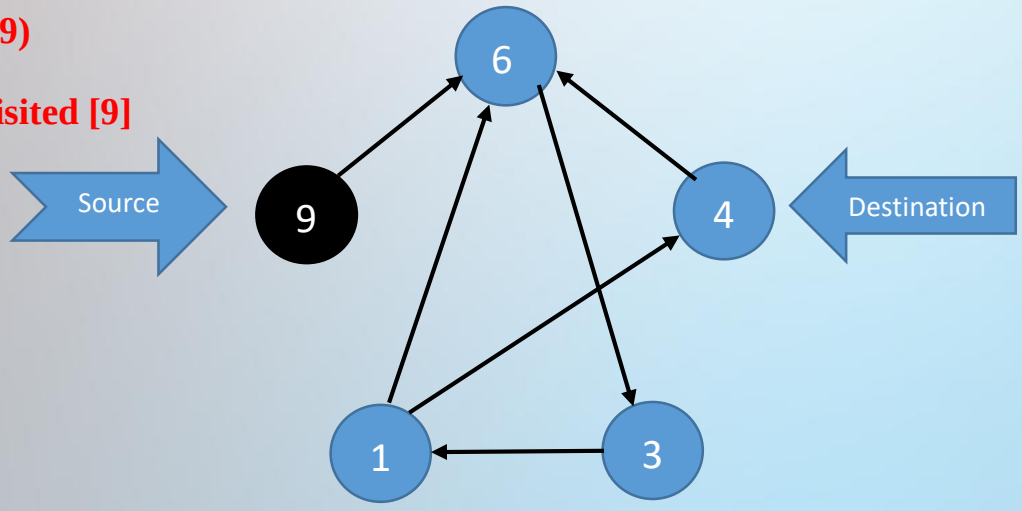
```
if (c == d)
	return true;
	bool *nodeVisited = new bool[value_element];
	for (int iterate = 0; iterate < value_element; iterate ++){
		nodeVisited[iterate] = false;
	}

	list<int> Quoo;
	nodeVisited[c] = true;
	Quoo.push_back(c);

	list<int>::iterator iterate;

	while (!Quoo.empty())
	{
		c = Quoo.front();
		Quoo.pop_front();
		for (iterate = adjaCent[c].begin(); iterate != adjaCent[c].end();
		++ iterate)
		{
			if (*iterate == d)
				return true;
			if (!nodeVisited[*iterate])
			{
				nodeVisited[*iterate] = true;
				Quoo.push_back(*iterate);
			}
		}
	}
	return false;
```

--Diagram—

First Step:

Quoo (9)

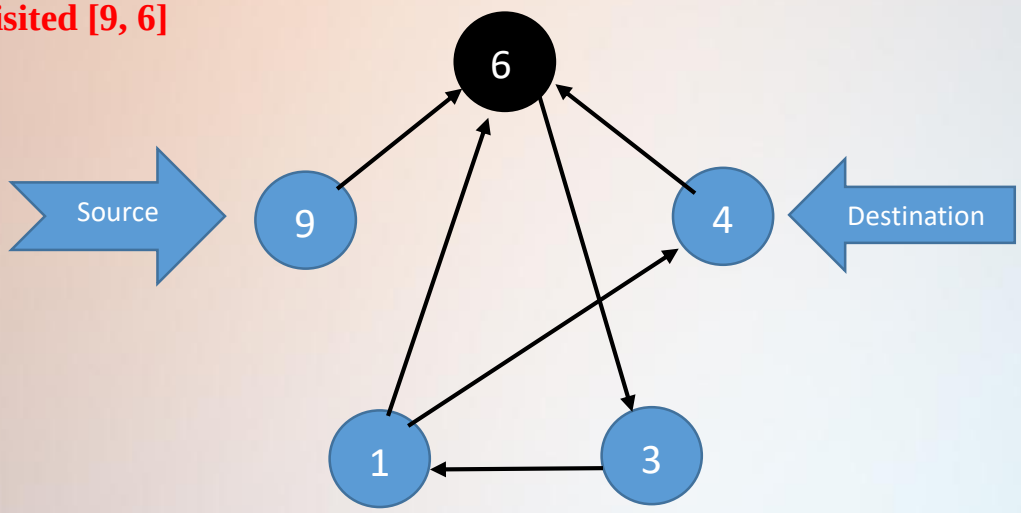
nodeVisited [9]



Second Step:

Quoo (6)

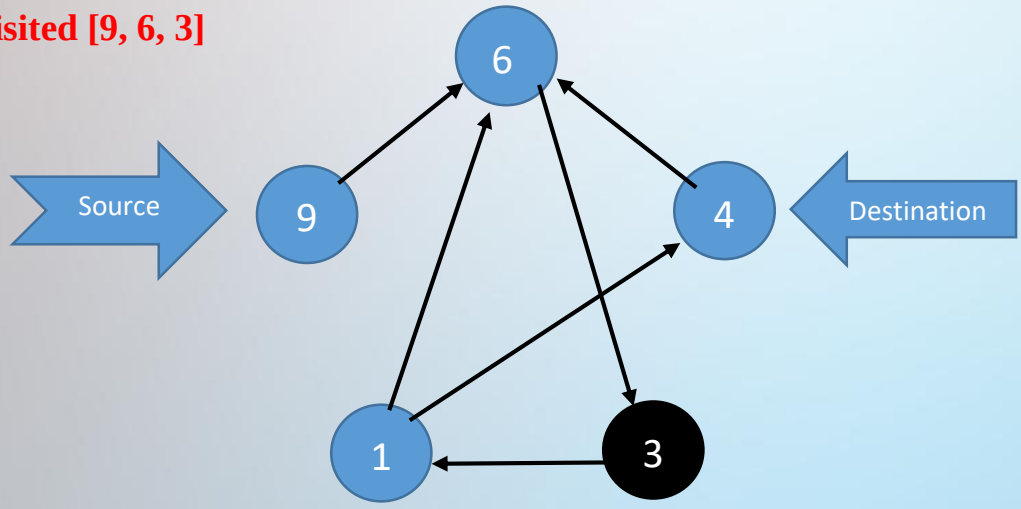
nodeVisited [9, 6]



Third Step:

Quoo (3)

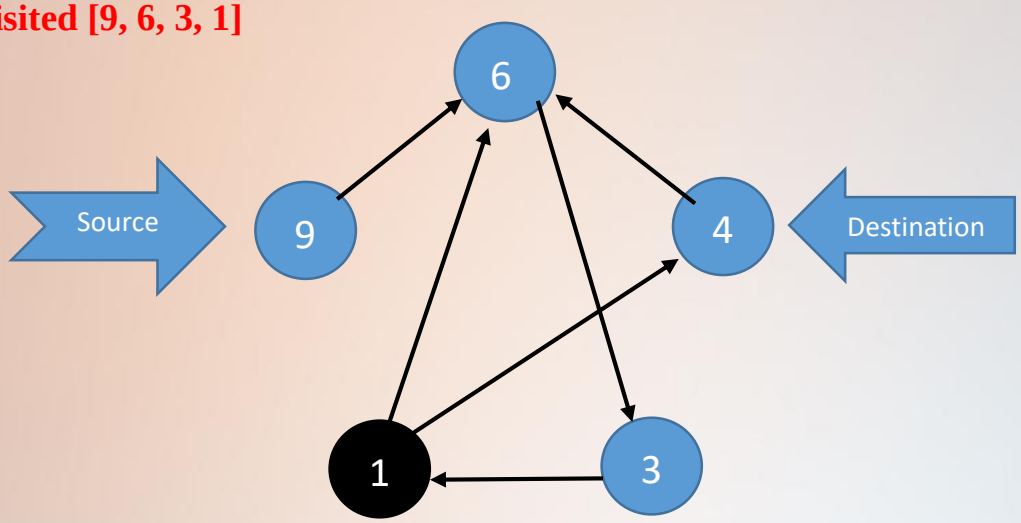
nodeVisited [9, 6, 3]



Fourth Step:

Quoo (1)

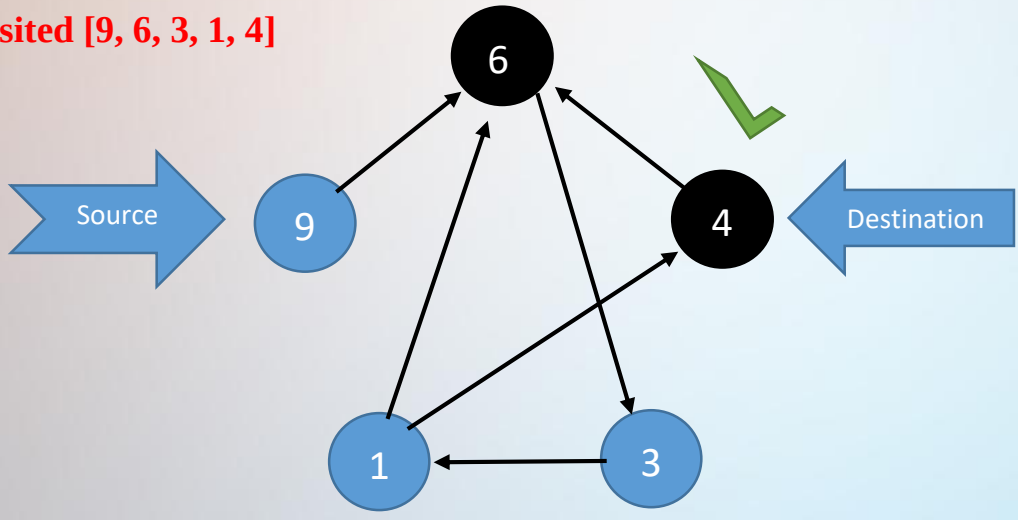
nodeVisited [9, 6, 3, 1]



Fifth Step:

Quoo (4, 6)

nodeVisited [9, 6, 3, 1, 4]



Since our destination (4) turned true inside the node visited list, we can say there is a path between targeted vertices which are 9 and 4.

Source: 6 Destination: 8

Quoo

nodeVisited

Vertices	9	6	3	1	4
Visited	No	No	No	No	No

First Step: Starts visiting from 9.

Quoo

9				

nodeVisited

Vertices	9	6	3	1	4
Visited	Yes	No	No	No	No

Second Step: Then it goes forward as directed to 6.

Quoo

6				

nodeVisited

Vertices	9	6	3	1	4
Visited	Yes	Yes	No	No	No

Third Step: Then we the 6 gets removes from queue, and the next vertices gets added which is 3.

Quoo

3				

nodeVisited

Vertices	9	6	3	1	4
Visited	Yes	Yes	Yes	No	No

Fourth Step: Then we the 3 gets removes from queue, and the next vertices gets added which is 1.

Quoo

1				

nodeVisited

Vertices	9	6	3	1	4
Visited	Yes	Yes	Yes	Yes	No

Fifth Step: Since 1 has a path to both 4 and 6, both of them get added to queue, and only 4 goes to visited list because 6 has already been visited and is in the list.

Quoo

4, 6				

nodeVisited

Vertices	9	6	3	1	4
Visited	Yes	Yes	Yes	Yes	Yes

Time Complexity for Question 6:

We use this formula to calculate the time complexity of directed graph: O(V+E).

V -> number of vertices E -> number of edges

Time complexity = O(V + E)

= O(5 + 6)

= O(11)

Space Complexity for Question 6:

We use this formula to calculate space complexity: O(V)

Which shows how many elements can be placed in a queue. So we can only have V number of elements in a queue.

Space Complexity = O(V)

= O(5)

C++ Codes for Question 6:

Bushra Naeemi\Desktop\Bushra Major Assignment\Question6.cpp - Dev-C++ 5.11

Search View Project Execute Tools AStyle Window Help

(globals)

Question6.cpp

```cpp
#include<iostream>
#include <list>
using namespace std;
//-------------------class---------------
class graphDirected
{
    int value_element;
    list<int> *adjaCent;
public:
    graphDirected(int value_element);
    void add_value_element(int a, int b);
    bool con_check(int c, int d);
};
//----------------method------------------
graphDirected::graphDirected(int value_element)
{
    this->value_element = value_element;
    adjaCent = new list<int>[value_element];
}
//------------------For Adding Value----------
void graphDirected::add_value_element(int a, int b)
{
    adjaCent[a].push_back(b);
}
```

```cpp
//------------------For Adding Value----------
void graphDirected::add_value_element(int a, int b)
{
    adjaCent[a].push_back(b);
}
//------------------For Checking connection---------
bool graphDirected::con_check(int c, int d)
{
    if (c == d)
    return true;
    bool *nodeVisited = new bool[value_element];
    for (int i = 0; i < value_element; i++){
        nodeVisited[i] = false;
    }

    list<int> Quoo;
    nodeVisited[c] = true;
    Quoo.push_back(c);

    list<int>::iterator i;

    while (!Quoo.empty())
    {
        c = Quoo.front();
        Quoo.pop_front();
        for (i = adjaCent[c].begin(); i != adjaCent[c].end(); ++i)
        {
            if (*i == d)
                return true;
            if (!nodeVisited[*i])
            {
                nodeVisited[*i] = true;
                Quoo.push_back(*i);
            }
        }
    }
    return false;
}
```

```cpp
//---------------Main method to test and run the codes----------
int main()
{
    graphDirected graph_DirecteD(10);
    graph_DirecteD.add_value_element(9, 6);
    graph_DirecteD.add_value_element(6, 3);
    graph_DirecteD.add_value_element(3, 1);
    graph_DirecteD.add_value_element(1, 6);
    graph_DirecteD.add_value_element(1, 4);

    int first = 9, second = 4;
    if(graph_DirecteD.con_check(first, second))
        cout<<endl<< "Path Exists from " << first << " --> " << second;
    else
        cout<<endl<< "Path doesn't Exist from " << first << " --> " << second;

    return 0;
}
```

You can also run the code from file through Question6.cpp.

--Output of Question 6—

```
C:\Users\Bushra Naeemi\Desktop\Bushra Major Assignment\Question6.exe

Path Exists from 9 --> 4
--------------------------------
Process exited after 0.04776 seconds with return value 0
Press any key to continue . . .
```

--Thank You--