Lab 6: Linked Queue & Linked Stack

1. In **main( )** function, the program prompt for the total number of integers to push into the queue, and continues to prompt input from the user for each integer and append each integer into the queue. Create a function named **traverse( )** in the class **ADTqueue** to trace and display the integers from the queue. The program will then serve each integer from the queue and calculates the sum of all the integers.
(Modify the full program in Chapter 5 on linked queue)

Sample output:

```
How many integers you want to append into the queue? 6

Enter integer 1 to the queue : 2
Enter integer 2 to the queue : 4
Enter integer 3 to the queue : 6
Enter integer 4 to the queue : 5
Enter integer 5 to the queue : 3
Enter integer 6 to the queue : 10

The integers that were append onto the queue are:
2 4 6 5 3 10

The sum of the integers is :
30
Press any key to continue
```

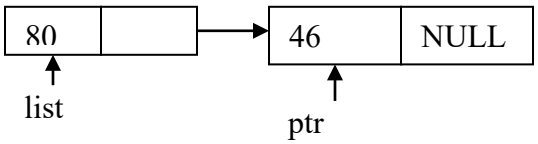
ANS:

```
#include<iostream>

using namespace std;

struct node
{
   int data;
   node *next;
};

class ADTqueue
{
   private:
      node *front, *rear;

   public:
      ADTqueue()
      {
         front=NULL;
         rear=NULL;
      }
      int empty()
      {
```

```
    if(front==NULL)
        return 1;
    else
        return 0;
}
void append(int num)
{
    if(rear!=NULL)
    {
        rear->next=new node;
        rear=rear->next;
        rear->data=num;
        rear->next=NULL;
    }
    else
    {
        front=rear=new node;
        front->data=num;
        front->next=NULL;
    }
}
int serve()
{
    int num;

    if(!empty())
    {
        num=front->data;
        node *temp=front;
        front=front->next;
        delete temp;

        if(front==NULL)
            rear=NULL;

        return num;
    }
    else
    {
        cout<<"Queue is empty";
        return 0;
    }

}
void traverse()
{
```

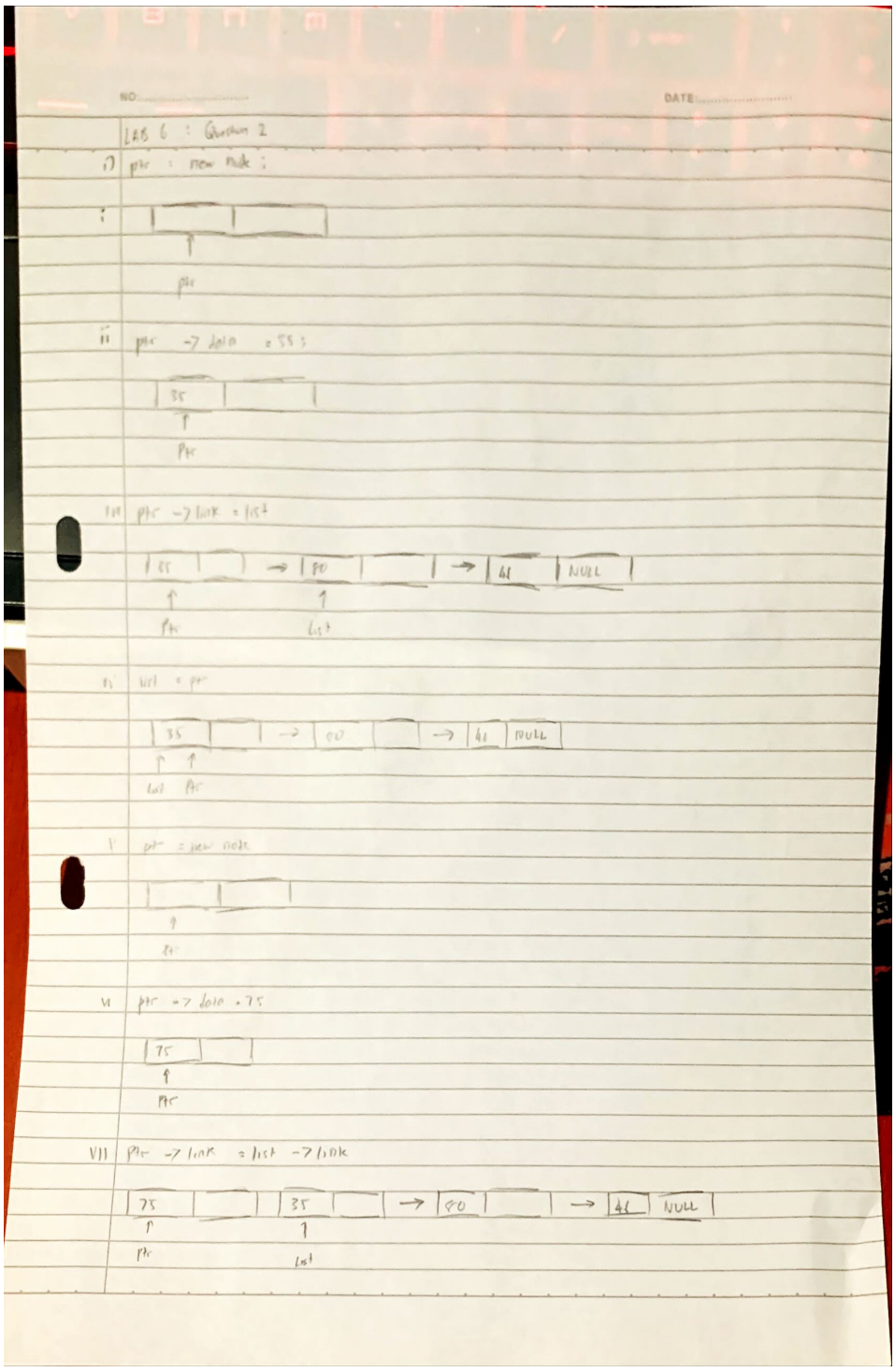
```
        node *temp;
        temp=front;

        while(temp!=NULL)
        {
           cout<<temp->data <<" ";
           temp=temp->next;
        }
     }
};

int main()
{
   ADTqueue q;
   int n, num, sum=0;

   cout<<"How many integers you want to append into the queue? ";
   cin>>n;
   cout<<endl;

   for(int i=0;i<n;i++)
   {
      cout<<"Enter integer "<<i+1<<" to the queue : ";
      cin>>num;
      q.append(num);
   }

   cout<<"\nThe integers that were append onto the queue are : \n";

   q.traverse();

   cout<<"\n\nThe sum of the integers is : "<<endl;

   while(!q.empty())
   {
      sum+=q.serve();
   }
   cout<<sum<<endl;

   system("pause");
   return 0;

}
```

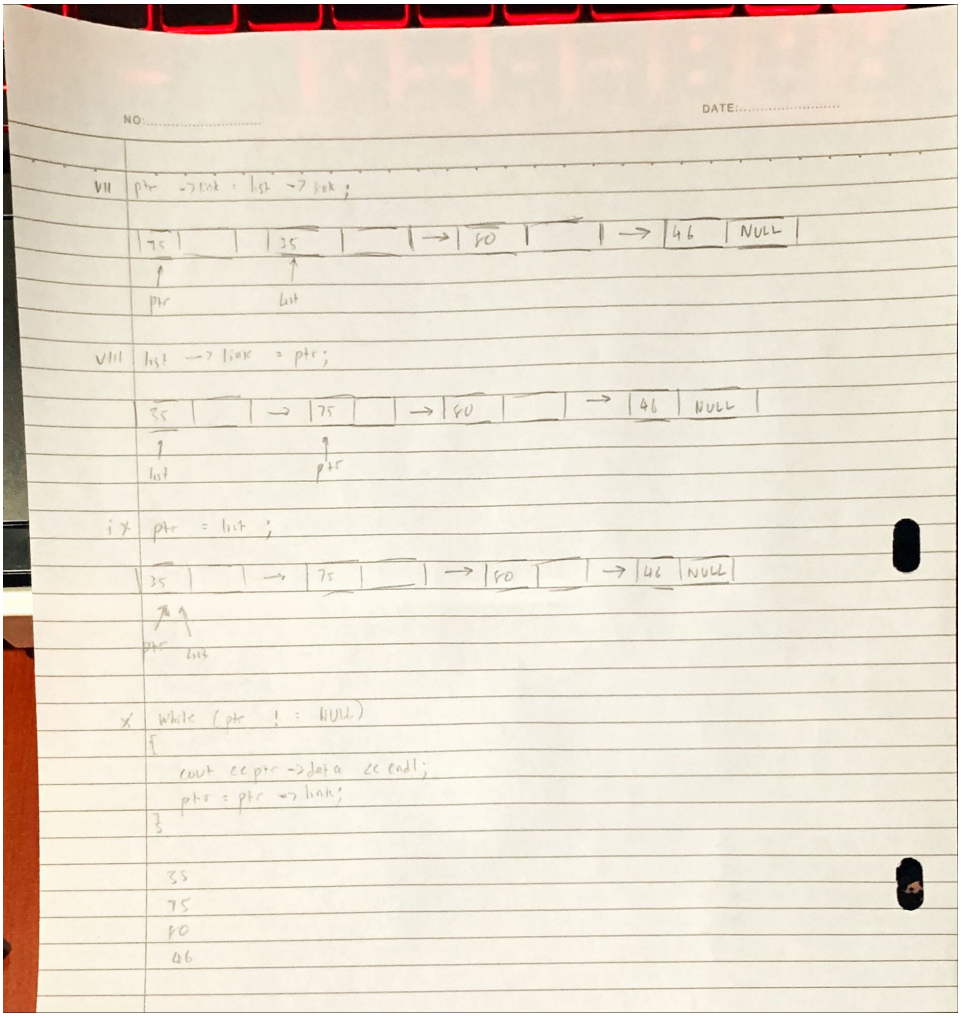
2. Given below is the current linked node that consists of data 80 and 46. Show and draw the trace diagram continued from the given diagram based on the following C++ codes. Assume that the node consists of two members, data and link with the data of the type int, (list and ptr are pointers of the type node).




```
struct node
{
   int data;
   node *link;
}*list, *ptr;

   i.    ptr = new node;
   ii.   ptr -> data = 35;
   iii.  ptr -> link = list;
   iv.   list = ptr;
   v.    ptr = new node;
   vi.   ptr -> data = 75;
   vii.  ptr -> link = list -> link;
   viii. list -> link = ptr;
   ix.   ptr = list;
   x.    while (ptr != NULL)
   {
         cout<<ptr->data<<endl;
         ptr = ptr -> link;
   }
```

ANS


NO.
DATE
LAB 6 : Question 2
i) ptr = new node;
ptr
ii) ptr -> data = 35;
35
ptr
iii) ptr -> link = list
35
40
41
NULL
ptr
list
iv) list = ptr
35
40
41
NULL
list
ptr
v) ptr = new node;
ptr
vi) ptr -> data = 75;
75
ptr
vii) ptr -> link = list -> link
75
35
40
41
NULL
ptr
list


NO:
DATE:
VII
ptr -> link = list -> link;
75
35
50
46
NULL
ptr
list
VIII
list -> link = ptr;
35
75
50
46
NULL
list
ptr
IX
ptr = list;
35
75
50
46
NULL
ptr
list
X
while (ptr != NULL)
{
cout << ptr -> data << endl;
ptr = ptr -> link;
}
35
75
50
46

Answer TRUE/FALSE for the following questions.

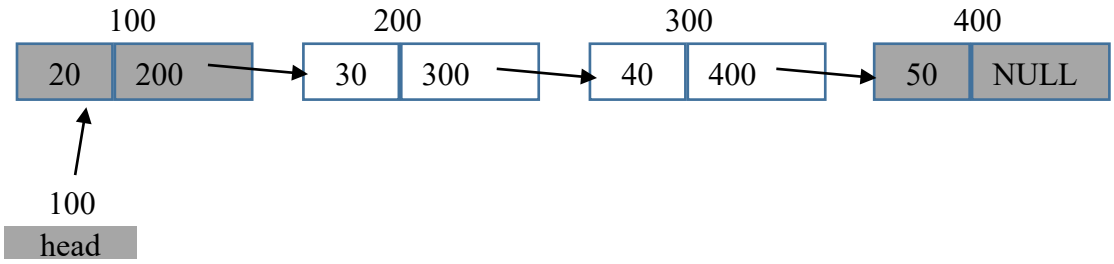
1. The process to add an element into a stack is known as popping.
   - False

2. If the following elements are append in this sequence in a queue: X,Y,Z, the order in which they are served are : Z,Y,X.
   - False

3. A linked list can be created using two ways : backward manner and forward manner.
   - True

4. One pointer must always point to the first node in a linked list.
   - True

5. Based on the following figure the value for head -> next -> next-> data is 40.
   - False



6. If the top pointer in a linked stack is NULL, the stack is empty.
   - True

7. We do not need to check whether the stack is full before pushing an element into the stack in a pointer based implementation.
   - True

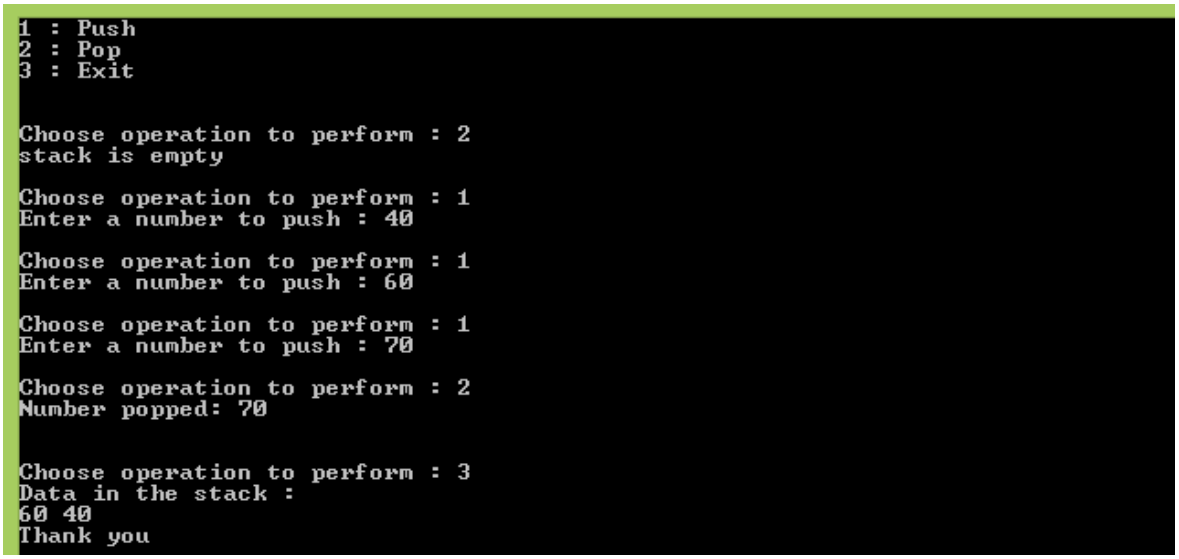
8. In a linked list implementation of a queue, an element can be served either at the front or at the back of the queue.
   - False

Submission question

Write a program to perform operation push and pop on a linked stack. The program will prompt the user for input and will display the operation result accordingly :

1 : Push – program prompt user input for number to be pushed into the linked stack.
2 : Pop – program will remove one element from the linked stack.
3 : Exit – program will display all the element in the linked stack and exit the system.
(Modify the full program in Chapter 5 on linked stack.)2

Sample output :



ANS:

```
#include <iostream>

using namespace std;

struct node
{
   int data;
   node *next;
};

class ADTstack
{
   private:
      node *top;
```

```
public:
   ADTstack()
   {
      top = NULL;
   }
   int empty()
   {
      if(top == NULL)
         return 1;
      else
         return 0;
   }
   void push(int num)
   {
      node *temp;
      temp = new node;
      temp->data = num;

      if(top == NULL)
      {
         top = temp;
         temp->next = NULL;
      }
      else
      {
         temp->next = top;
         top = temp;
      }
   }
   int pop()
   {
      int num=0;
      node *temp;

      if(!empty())
      {
         num = top->data;
         temp = top;
         top = top->next;
         delete temp;
         return num;
      }
      else
      {
         cout<<"Stack is Empty\n";
         return 0;
```

```
        }

    }
    void display()
    {
        node* temp = top;
        cout<<"Data in the stack : "<<endl;

        while(temp != NULL)
        {
            cout<<temp->data<<" ";
            temp=temp->next;
        }
        cout<<endl;
    }
};

int main()
{
    ADTstack st;

    int ans, number;

    cout<<"1 : Push"<<endl;
    cout<<"2 : Pop"<<endl;
    cout<<"3 : Exit"<<endl;

    int choice, value;

    do{
        cout<<"\nEnter Operation to perform: ";
        cin>>choice;

        if(choice==1)
        {
            cout<<"Enter value to push: ";
            cin>>value;
            st.push(value);
        }
        else if(choice==2)
        {
            if(!st.empty())
            {
                cout<<"Number popped: "<<st.pop()<<endl;
            }
            else
```

```
            {
               st.pop();
            }
         }
         else if(choice==3)
         {
            break;
         }
         else
         {
            cout<<"Invalid choice"<<endl;
         }
   }while(choice!=3);

   st.display();

   system("pause");
   return 0;
}
```

Ouput:

```
1 : Push
2 : Pop
3 : Exit

Enter Operation to perform: 2
Stack is Empty

Enter Operation to perform: 1
Enter value to push: 40

Enter Operation to perform: 1
Enter value to push: 60

Enter Operation to perform: 1
Enter value to push: 70

Enter Operation to perform: 2
Number popped: 70

Enter Operation to perform: 3
Data in the stack :
60 40
Press any key to continue . . .
```