DS LAB-A WEEK-2

-20103153_AVNI_B6

Q1.

```
#include<iostream>
using namespace std;

class Node
{
public:
    int data;
    Node *next;

    Node(int info)
    {
        data=info;
        next=NULL;
    }

};

class List
{
public:
    Node *head;
    int c;
    Node *tail;
    List()
    {
        head=NULL;
        c=0;
        tail=NULL;
    }
    List* createList()
    {

        List *new1=new List;
        return new1;
    }
    void insert(int val)
    {
      Node * new_node = new Node(val);

      if (head == NULL)
        head = new_node;

      else
      {
        new_node->next = head;
        head = new_node;
      }
    }

    void display()
    {
      Node* temp = head;
      while(temp != NULL)
      {
        cout << temp->data << " ";
        temp = temp->next;
      }
      cout << endl;
    }
};
int main() {

  List l;
  // inserting elements
  l.insert(6);
  l.insert(9);
  l.insert(1);
  l.insert(3);
  l.insert(7);
  l.insert(6);
  l.insert(9);
  l.insert(1);
  l.insert(3);
  l.insert(7);
  cout << "Current Linked List: ";
  l.display();
}
```

```
Current Linked List: 7 3 1 9 6 7 3 1 9 6

Process returned 0 (0x0)   execution time
Press any key to continue.
```

Q2.

```
#include<iostream>
using namespace std;

class Node
{
public:
    int data;
    Node *next;

    Node(int info)
    {
        data=info;
```

```
        next=NULL;
    }

};

class List
{
public:
    Node *head;
    int c;
    Node *tail;
    List()
    {
        head=NULL;
        c=0;
        tail=NULL;
    }
    List* createList()
    {

        List *new1=new List;
        return new1;
    }
    void insert(int val)
    {
      Node * new_node = new Node(val);

      if (head == NULL)
        head = new_node;

      else
      {
        new_node->next = head;
        head = new_node;
      }
    }

    void count()
    {
      Node* temp = head;
      int mn=INT_MAX,mx=INT_MIN,c=0;

      while(temp != NULL)
      {
        mn=min( temp->data,mn);
        mx=max( temp->data,mx);
        c++;
        temp = temp->next;
      }
      cout<<"TOTAL number of nodes=
"<<c<<endl;
      cout << "MAXIMUM VALUE= "<<mx<<"
and MINIMUM VALUE= "<<mn<<endl;
    }
};
```

```
int main() {

  List l;
  // inserting elements
  l.insert(6);
  l.insert(9);
  l.insert(1);
  l.insert(3);
  l.insert(7);
  l.insert(6);
  l.insert(9);
  l.insert(1);
  l.insert(3);
  l.insert(7);
  l.count();
}
```

```
TOTAL number of nodes= 10
MAXIMUM VALUE= 9 and MINIMUM VALUE= 1

Process returned 0 (0x0)   execution time : 0.387 s
Press any key to continue.
```

Q3.

```
#include<iostream>
using namespace std;

class Node
{
public:
    int data;
    Node *next;

    Node(int info)
    {
        data=info;
        next=NULL;
    }

};

class List
{
public:
    Node *head;
    int c;
    Node *tail;
    List()
    {
        head=NULL;
        c=0;
        tail=NULL;
    }
```

```
    List* createList()
    {

        List *new1=new List;
        return new1;
    }
    void insertAtBeginning(int val)
    {
      Node * new_node = new Node(val);

      if (head == NULL)
        head = new_node;

      else
      {
        new_node->next = head;
        head = new_node;
      }
    }

    void display()
    {
      Node* temp = head;
      while(temp != NULL)
      {
        cout << temp->data << " ";
        temp = temp->next;
      }
      cout << endl;
    }
};
int main() {

  List l;
  // inserting elements
  l.insertAtBeginning(6);
  l.insertAtBeginning(9);
  l.insertAtBeginning(1);
  l.insertAtBeginning(3);
  l.insertAtBeginning(1);
  l.insertAtBeginning(3);
cout << "Current Linked List: ";
  l.display();
  cout << "Inserting 7 at the Beginning
of the Linked List: ";
  l.insertAtBeginning(7);
  l.display();
}
```

```
Current Linked List: 3 1 3 1 9 6
Inserting 7 at the Beginning of the Linked List: 7 3 1 3 1 9

Process returned 0 (0x0)   execution time : 4.382 s
Press any key to continue.
```

Q4.

```
#include<iostream>
using namespace std;

class Node
{
public:
    int data;
    Node *next;

    Node(int info)
    {
        data=info;
        next=NULL;
    }

};

class List
{
public:
    Node *head;
    int c;
    Node *tail;
    List()
    {
        head=NULL;
        c=0;
        tail=NULL;
    }
    List* createList()
    {

        List *new1=new List;
        return new1;
    }
    void insertAtLocation(int val,int
pos)
    {
      Node * new_node = new
Node(val),*temp;
      temp=head;
      if(pos==1)
      {
           new_node->next = head;
           head = new_node;

      }
      else{
      pos--;
      while(--pos)
      {
          temp=temp->next;
      }
      new_node->next=temp->next;
      temp->next=new_node;
      }
```

```
    }
    void insert(int val)
    {
      Node * new_node = new Node(val);

      if (head == NULL)
        head = new_node;

      else
      {
        new_node->next = head;
        head = new_node;
      }
    }

    void display()
    {
      Node* temp = head;
      while(temp != NULL)
      {
        cout << temp->data << " ";
        temp = temp->next;
      }
      cout << endl;
    }
};
int main() {

  List l;
  // inserting elements
  l.insert(6);
  l.insert(9);
  l.insert(1);
  l.insert(3);
  l.insert(7);
  cout << "Current Linked List: ";
  l.display();
  cout << "Inserting 8 at position 2 of
Linked List: ";

  l.insertAtLocation(8,2);
  l.display();

}
```

```
Current Linked List: 7 3 1 9 6
Inserting 8 at position 2 of Linked List: 7 8 3 1 9 6

Process returned 0 (0x0)   execution time : 0.324 s
Press any key to continue.
```

Q5.

```
#include<iostream>
using namespace std;

class Node
{
public:
    int data;
    Node *next;

    Node(int info)
    {
        data=info;
        next=NULL;
    }

};

class List
{
public:
    Node *head;
    int c;
    Node *tail;
    List()
    {
        head=NULL;
        c=0;
        tail=NULL;
    }
    List* createList()
    {

        List *new1=new List;
        return new1;
    }
    void insert(int val)
    {
      Node * new_node = new Node(val);

      if (head == NULL)
        head = new_node;

      else
      {
        new_node->next = head;
        head = new_node;
      }
    }

    void display()
    {
      Node* temp = head;
      while(temp != NULL)
      {
        cout << temp->data << " ";
        temp = temp->next;
      }
      cout << endl;
    }
```

```
};
int main() {

  List l;
  int n;
  // inserting elements
  cout<<"Enter any number : ";
  cin>>n;
  while(n!=0)
  {
      l.insert(n%10);
      n=n/10;
  }
  cout << "Current Linked List: ";
  l.display();
}
```

```
Enter any number : 13651434
Current Linked List: 1 3 6 5 1 4 3 4

Process returned 0 (0x0)   execution time : 4.026 s
Press any key to continue.
```

Q6.

```
#include<iostream>
using namespace std;

class Node
{
public:
    char data;
    Node *next;

    Node(char info)
    {

        data=info;
        next=NULL;
    }

};

class List
{
public:
    Node *head;
    int c;
    Node *tail;
    List()
    {
        head=NULL;
        c=0;
        tail=NULL;
    }
    List* createList()
    {
        List *new1=new List;
        return new1;
    }
    void insert(char val)
    {
      Node * new_node = new Node(val);

      if (head == NULL)
        {head = new_node;
         tail=new_node;
        }

      else
      {
        tail->next = new_node;
        tail = new_node;
      }
    }

    void display()
    {
      Node* temp = head;
      while(temp != NULL)
      {
        cout << temp->data << " ";
        temp = temp->next;
      }
      cout << endl;
    }
};
int main() {

  List l;
  string n;
  // inserting elements
  cout<<"Enter any Name : ";
  cin>>n;
  for(int i=0;i<n.size();i++)
  {
      l.insert(n[i]);

  }
  cout << "Current Linked List: ";
  l.display();
}
```

```
Enter any Name : AVNI
Current Linked List: A V N I

Process returned 0 (0x0)   execution time : 4.807 s
Press any key to continue.
```

Q7.

```
#include<iostream>
using namespace std;

class Node
{
public:
    char data;
    Node *next;

    Node(char info)
    {

        data=info;
        next=NULL;
    }

};

class List
{
public:
    Node *head;
    int c;
    Node *tail;
    List()
    {
        head=NULL;
        c=0;
        tail=NULL;
    }
    List* createList()
    {

        List *new1=new List;
        return new1;
    }
    void insert(char val)
    {
      Node * new_node = new Node(val);

      if (head == NULL)
        {head = new_node;
         tail=new_node;
        }

      else
      {
        tail->next = new_node;
        tail = new_node;
      }
    }

    void remove(List l2)
    {
      Node* temp = head;
      while(temp != NULL)
      {
        Node* temp1 = l2.head;


        for(int i=0;i<3;i++)
        {
            if(temp->data==temp1->data
&& temp->next->data==temp1->next->data
&& temp->next->next->data==temp1->next-
>next->data)
            {
                        Node* curr =
head;
                        Node* prev =
head;
                        while(curr
!=NULL && curr->data!=temp->data)
                            {
                            prev = curr;
                            curr = curr-
>next;
                            }

                        prev->next=temp-
>next->next->next;

            }

            temp1=temp1->next;
        }

        temp = temp->next;
      }
     display();
    }
     void display()
    {
      Node* temp = head;
      while(temp != NULL)
      {
        cout << temp->data << " ";
        temp = temp->next;
      }
      cout << endl;
    }
};
int main() {

  List l1,l2;
  string s1="abcdefghij",s2="defab";

  for(int i=0;i<s1.size();i++)
  {
```

```
      l1.insert(s1[i]);

  }
  for(int i=0;i<s2.size();i++)
  {
      l2.insert(s2[i]);

  }
  cout << "Current Linked List - 1 : ";
  l1.display();
  cout << "Current Linked List - 2 : ";
  l2.display();
  cout << "After calling remove function
Linked List 1 becomes : ";
  l1.remove(l2);

}
```

```
Current Linked List - 1 : a b c d e f g h i j
Current Linked List - 2 : d e f a b
After calling remove function Linked List 1 becomes : a b c g h i

Process returned 0 (0x0)   execution time : 0.992 s
Press any key to continue.
```

Q8.

```
The list contains: 10 20 30
Inserting an 100 at position 2
The list contains: 10 100 20 30
Inserting an 200 at position 1
The list contains: 200 10 100 20 30

Process returned 0 (0x0)   execution
Press any key to continue.
```

```
#include <iostream>
using namespace std;

struct Node {
    int data;
    Node* next;
    Node* prev;
};

class LinkedList {
  private:
    Node* head;
  public:
    LinkedList(){
      head = NULL;
    }

    void push_back(int newElement) {
      Node* newNode = new Node();
      newNode->data = newElement;
      newNode->next = NULL;
      newNode->prev = NULL;
      if(head == NULL) {
        head = newNode;
      } else {
        Node* temp = head;
        while(temp->next != NULL)
          temp = temp->next;
        temp->next = newNode;
        newNode->prev = temp;
      }
    }

    void push_at(int newElement, int
position) {
      Node* newNode = new Node();
      newNode->data = newElement;
      newNode->next = NULL;
      newNode->prev = NULL;
      if(position < 1) {
        cout<<"\nposition should be >=
1.";
      } else if (position == 1) {
        newNode->next = head;
        head->prev = newNode;
        head = newNode;
      } else {
        Node* temp = head;
        for(int i = 1; i < position-1;
i++) {
          if(temp != NULL) {
            temp = temp->next;
          }
        }
        if(temp != NULL) {
          newNode->next = temp->next;
          newNode->prev = temp;
          temp->next = newNode;
          if(newNode->next != NULL)
            newNode->next->prev =
newNode;
        } else {
          cout<<"\nThe previous node is
null.";
        }
      }
    }

    void PrintList() {
      Node* temp = head;
      if(temp != NULL) {
        cout<<"The list contains: ";
        while(temp != NULL) {
          cout<<temp->data<<" ";
          temp = temp->next;
```

```
        }
        cout<<endl;
      } else {
        cout<<"The list is empty.\n";
      }
    }
};

int main() {
  LinkedList MyList;

  MyList.push_back(10);
  MyList.push_back(20);
  MyList.push_back(30);
  MyList.PrintList();

  cout<<"Inserting an 100 at position
2"<<endl;
  MyList.push_at(100, 2);
  MyList.PrintList();

  cout<<"Inserting an 200 at position
1"<<endl;
  MyList.push_at(200, 1);
  MyList.PrintList();

  return 0;
}
```

Q9.

```
#include <iostream>
using namespace std;

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


Node* removeLastNode(struct Node* head)
{
      if (head == NULL)
            return NULL;

      if (head->next == NULL) {
            delete head;
            return NULL;
      }

      Node* second_last = head;
      while (second_last->next->next !=
NULL)
            second_last = second_last-
>next;

      delete (second_last->next);
      second_last->next = NULL;

      return head;
}

void push(struct Node** head_ref, int
new_data)
{
      struct Node* new_node = new Node;
      new_node->data = new_data;
      new_node->next = (*head_ref);
      (*head_ref) = new_node;
}

int main()
{
      Node* head = NULL;
      push(&head, 12);
      push(&head, 29);
      push(&head, 11);
      push(&head, 23);
      push(&head, 8);
      cout<<"Currently List is: ";
    for (Node* temp = head; temp !=
NULL; temp = temp->next)
            cout << temp->data << " ";
      head = removeLastNode(head);
      cout<<endl<<"After removing last
element The list becomes: ";
      for (Node* temp = head; temp !=
NULL; temp = temp->next)
            cout << temp->data << " ";

      return 0;
}
```

```
Currently List is: 8 23 11 29 12
After removing last element The list becomes: 8 23 11 29
Process returned 0 (0x0)   execution time : 0.955 s
Press any key to continue.
```

Q10.

```
#include<iostream>
using namespace std;
string
name[4]={"first","second","third","fourt
h"};
class Node
{
public:
    int data;
    Node *next;
    Node *prev;

    Node(int info)
    {
```

```
        data=info;
        next=NULL;
        prev=NULL;
    }

};

class List
{
public:
    Node *head;
    int c=0;
    Node *tail;
    List()
    {
        head=NULL;
        c=0;
        tail=NULL;
    }
    List* createList()
    {

        List *new1=new List;
        return new1;
    }
    void insert(int val)
    {
        c++;
            Node* new_node = new
Node(val);
            new_node->next = head;

            if (head != NULL)
               {
                   head->prev =
new_node;
                   head = new_node;
               }
               else{
                head=new_node;
                tail=new_node;
               }
    }

    void display()
    {
      Node* temp = head;
      while(temp != NULL)
      {
        cout << temp->data << " ";
        temp = temp->next;
      }
      cout << endl;
    }

    void extremeSwap()
    {
        Node* t1 = head;
        Node* t2 = tail;

        for(int i=0;i<c/2;i++)
        {
            cout<<"Calling EXTREME Swap
function for "<<name[i]<<" time:"<<endl;
            swap(t1->data,t2->data);
            t1=t1->next;
            t2=t2->prev;
            display();
            cout<<endl;
        }


    }
};
int main() {

  List l;
  l.insert(1);
  l.insert(2);
  l.insert(3);
  l.insert(4);
  l.insert(5);
  l.insert(6);
  l.insert(7);
  l.insert(8);
  cout << "Current Linked List: ";
  l.display();
  l.extremeSwap();
}
```

```
Current Linked List: 8 7 6 5 4 3 2 1
Calling EXTREME Swap function for first time:
1 7 6 5 4 3 2 8

Calling EXTREME Swap function for second time:
1 2 6 5 4 3 7 8

Calling EXTREME Swap function for third time:
1 2 3 5 4 6 7 8

Calling EXTREME Swap function for fourth time:
1 2 3 4 5 6 7 8


Process returned 0 (0x0)   execution time : 2.956 s
Press any key to continue.
```