Assignment No 8

Second year Computer Engineering class, set A of students like Vanilla Ice-cream and set B of students like butterscotch ice-cream. Write C++ program to store two sets using linked list. Compute and display-

a) Set of students who like both vanilla and butterscotch

b) Set of students who like either vanilla or butterscotch or not both

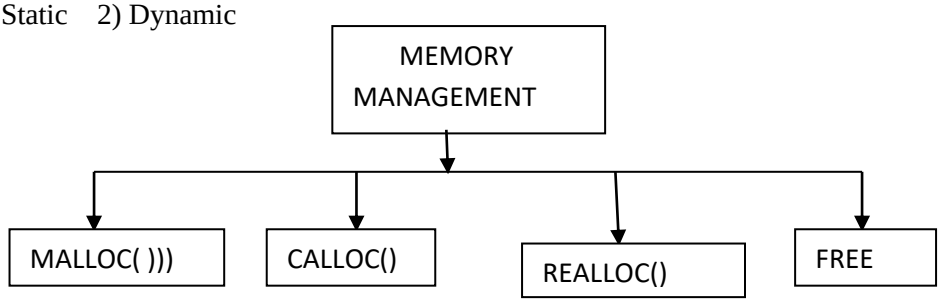
c) Number of students who like neither vanilla nor butterscotch

Theory

DYNAMIC MEMORY MANAGEMENT:-

In dynamic data structure, the memory space required by variables is calculated & allocated during execution. C gives us two choices for reserving memory locations for a variable-

1) Static 2) Dynamic



1.] Block Memory Allocation (malloc()):

The malloc() function allocates a block of memory that contains the no. of bytes specified in its parameter. This returns a pointer to an area of memory with size, byte-size.

2.] Contigous Memory Allocation (calloc()):

This function can be used to allocate a contiguous block of memory, primarily for arrays. It differs from malloc() only to the extent that it sets memory to null characters. This function returns a pointer to the allocated memory.

3.] Reallocation Of Memory (realloc()):

This function changes the size of the previously allocated block of memory. This is done by either deleting or extending the memory at the end of the block. If memory cannot be extended, realloc() allocates a completely new block of memory.

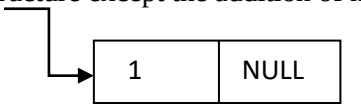
4.] Releasing Memory (free()):

When the allocated memory is no longer needed, it should be returned back by using the function free().

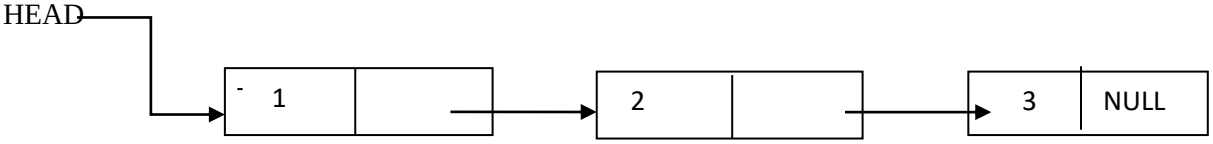
OPERATIONS ON LINK LIST:

1.CREATION:-

This operation is used to create constituent node as & when required. The structure of a list changes when nodes are inserted & deleted. Normally, creation of a list does not require alternation of the list structure except the addition of new node at the end or the beginning of the list.



Creating the first node



TAIL

The complete list

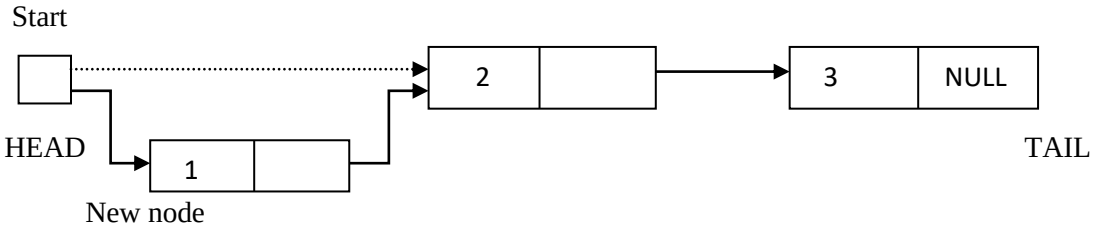
2.INSERTION :

This operation is used to insert a new node in the link list. This can be done by three ways as:-

(i)At the beginning of the list
(ii)At a certain position
(iii)At the end

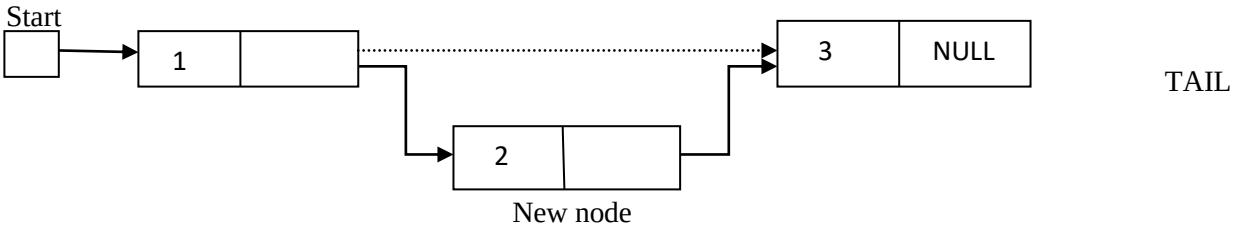
(i.)Insertion at the beginning of list:-

The insertion of a new node at the head of list is easier. The new node becomes the first node in the last i.e the address of the pointer head is changed to the address of new node.



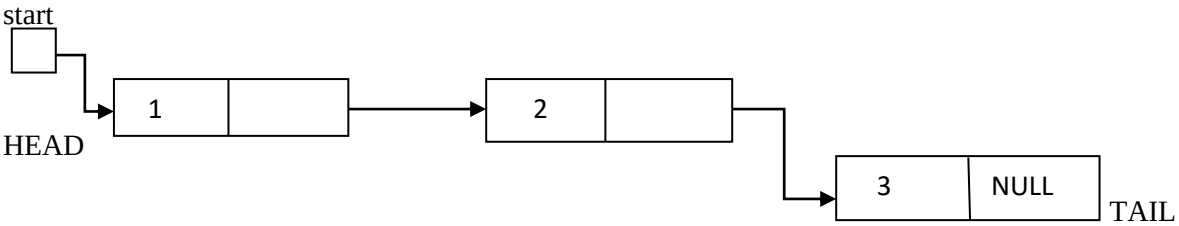
(ii.) Insertion at a certain position:-

The insertion of a data element at any point in data structure should be done in constant time. It requires two pointers to be changed. The node with data element 2 is to be inserted between data element 1 & 3. The link of new node is assigned the address of the node with data 3 & the link of the first node is changed to point to the new node.



(iii.)Insertion at the end of list:-

In the link list , only the head of the list is known to us. Therefore, to insert a node at the tail of the list, entire list has to be traversed. Once the last node is reached, the new node can be attached to the list.



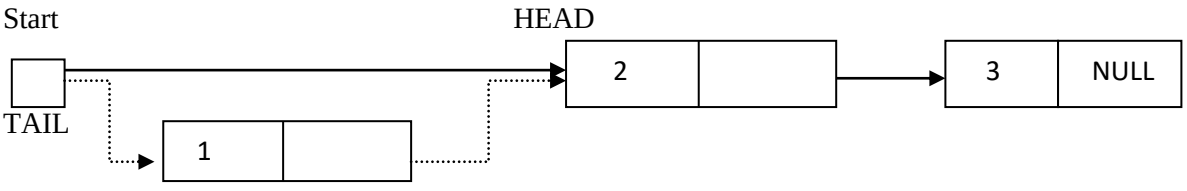
3.DELETION:-

This operation is used to delete node from the list. This can be done by three ways:-

(i)At the beginning of the list
(ii)At a certain position
(iii)At the end of list

(i.)Deletion at the beginning of the list:-

If the element to be deleted is located at the first position, the deletion procedure only consists of advancing the HEAD pointer to the second element in the list.

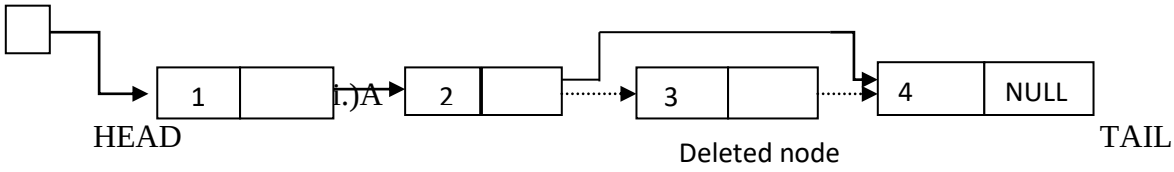


Delete node

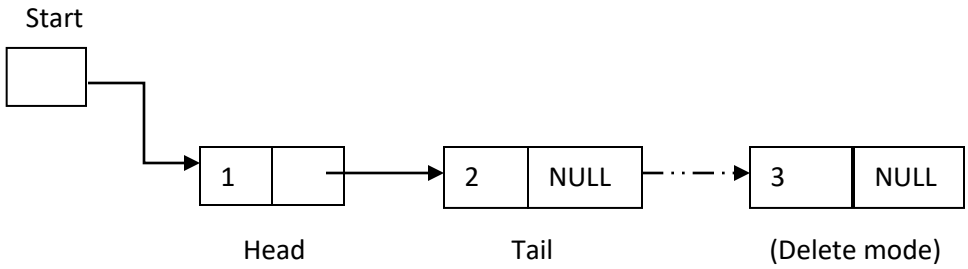
(ii.)At a certain position of the list:-

If an element in the middle of the list has to be deleted, only one pointer value has to be changed. The value in the link field of the preceding node is changed to the address of succeeding node.

Start



If the element to be deleted is the last node in the list, the link of the previous is made NULL
Thereby excluding the last element from the list.



4. **SEARCHING:-**

Searching refers to the finding of an item or data from a given list of elements .So in case of link list searching means finding a specific code from a specific list. In this operation we compare the node value to be searched with each node value in the list.

CONCLUSION:

Hence we have studied all the operations on link list successfully.

Problem statement: In SE Computer Engineering class set A of students like vanilla ice cream and set B of students like Butterscotch ice cream. Write C++ program to store two sets using linked list. Compute and display:

1. Set of students who like either vanilla or butterscotch or both
2. Set of students who like both vanilla and butterscotch
3. Set of students who like only vanilla but not butterscotch
4. Set of students who like only vanilla but not butterscotch
5. Number of students who like neither vanilla nor butterscotch

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

class node  //class name.
{
 public:
                int data;
                node *next; // to store of the next node.
};

node *create() //to create the list.
{
 int n,i;
 node *p, *head,*temp;
 head=NULL;
 cout<<"\n Enter the number of students:";
 cin>>n;
 for(i=0;i<n;i++)
  {
   p=(node*)malloc(sizeof(node));
   cout<<"\n Enter the roll number of students:";
   cin>>p->data;
   p->next=NULL;
   if(head==NULL)
    {
     head=p;
    }
```

```
   else
    {
     temp=head;
     while(temp->next!=NULL)
      {
        temp=temp->next;
      }
        temp->next=p;

     }

  }
 return head;
}

int unio(node *head,node *head1) //funtion for union of set.
{
 node *p ,*q;
 int f=0,count=0;
 p=head;
 while(p!=NULL)
  {
    cout<<"\t"<<p->data;
    p=p->next;
    count++;
  }
 for(q=head1;q!=NULL;q=q->next)
  {
    f=0;
    for(p=head;p!=NULL;p=p->next)
      {
        if(q->data==p->data)
          {
           f=1;
           break;
          }
      }
     if(f!=1)
      {
        cout<<"\t"<<q->data;
```

```
       count++;
      }

  }
 return count;
}

void inter(node *head1, node *head2)  //function for intersection of sets.
{
 node *p, *q;
 int f=0;
 for(q=head2;q!=NULL;q=q->next)
  {
   f=0;
   for(p=head1;p!=NULL;p=p->next)
    {
     if(q->data==p->data)
      {
       f=1;
       break;
      }
    }
  if(f==1)
   {
    cout<<"\t"<<q->data;
   }
  }
}

void vani(node *head1, node *head2) //function for only vanilla.
{
    node *p,*q;
    int f=0;

    for(p=head1;p!=NULL;p=p->next)
    {
        f=0;
              for(q=head2;q!=NULL;q=q->next)
              {
                 if(p->data==q->data)
```

```
                {
                   f=1;
                   break;
                }
             }


       if(f!=1)
       {
         cout<<"\t"<<p->data;

       }
    }
}

void butter(node *head1,node *head2) //function for only butterscotch
{
 node *p,*q;
  int f=0;

    for(q=head2;q!=NULL;q=q->next)
    {
       f=0;
             for(p=head1;p!=NULL;p=p->next)
             {
                if(p->data==q->data)
                {
                   f=1;
                   break;
                }
             }


       if(f!=1)
       {
          cout<<"\t"<<q->data;

       }
    }
}
```

```
int main()
{
 node *head, *head1;
 int choice,m,count=0;
 head=head1=NULL;
 cout<<"\n Enter the number of students:";
 cin>>m;
 head=NULL;

 while(choice!=7)
  {
   cout<<"\nEnter \n1]create \n2]students like either vanilla or butterscotch\n 3]students like both vanilla and butterscotch\n 4]students like vanilla only\n 5]students like only butterscotch\n 6]students like neither vanilla nor butterscotch";
   cout<<"\n Enter your choice:";
   cin>>choice;

   switch(choice)
    {
     case 1:
          cout<<"\n students like vanilla:";
          head=create();
          cout<<"\n students like butterscotch:";
          head1=create();
          break;
     case 2:
          cout<<"\n students like either vanilla or butterscotch:";
          count=unio(head,head1);
          break;
     case 3:
          cout<<"\n students like both vanilla and butterscotch:";
          inter(head,head1);
          break;
     case 4:
          cout<<"\n Students like vanilla only:";
          vani(head,head1);
          break;
     case 5:
          cout<<"\n students like butterscotch only:";
```

```
            butter(head,head1);
            break;
      case 6:
            count=unio(head,head1);
            cout<<"\n students like neither vanilla nor butterscotch:"<<m-count;
            break;
     }
   }
return 0;
}


/*
Output:
Enter the number of students:5

Enter
1]create
2]students like either vanilla or butterscotch
 3]students like both vanilla and butterscotch
 4]students like vanilla only
 5]students like only butterscotch
 6]students like neither vanilla nor butterscotch
 Enter your choice:1

 students like vanilla:
 Enter the number of students:2

 Enter the roll number of students:11

 Enter the roll number of students:22

 students like butterscotch:
 Enter the number of students:3

 Enter the roll number of students:11

 Enter the roll number of students:33

 Enter the roll number of students:44
```

Enter
1]create
2]students like either vanilla or butterscotch
3]students like both vanilla and butterscotch
4]students like vanilla only
5]students like only butterscotch
6]students like neither vanilla nor butterscotch
Enter your choice:2

students like either vanilla or butterscotch: 11 22 33 44
Enter
1]create
2]students like either vanilla or butterscotch
3]students like both vanilla and butterscotch
4]students like vanilla only
5]students like only butterscotch
6]students like neither vanilla nor butterscotch
Enter your choice:3

students like both vanilla and butterscotch: 11
Enter
1]create
2]students like either vanilla or butterscotch
3]students like both vanilla and butterscotch
4]students like vanilla only
5]students like only butterscotch
6]students like neither vanilla nor butterscotch
Enter your choice:4

Students like vanilla only: 22
Enter
1]create
2]students like either vanilla or butterscotch
3]students like both vanilla and butterscotch
4]students like vanilla only
5]students like only butterscotch
6]students like neither vanilla nor butterscotch
Enter your choice:5

students like butterscotch only: 33 44

Enter

1]create

2]students like either vanilla or butterscotch

3]students like both vanilla and butterscotch

4]students like vanilla only

5]students like only butterscotch

6]students like neither vanilla nor butterscotch

Enter your choice:6

11 22 33 44

students like neither vanilla nor butterscotch:1*/