**WAP to Implement Singly Linked List with following operations**
**a) Create a linked list.**
**b) Insertion of a node at first position, at any position and at end of list.**
**c) Display the contents of the linked list.**
**WAP to Implement Singly Linked List with following operations**
**a) Create a linked list.**
**b) Deletion of first element, specified element and last element in the list.**
**c) Display the contents of the linked list.**

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<process.h>

struct node
{
  int info;
  struct node *link;
};

typedef struct node *NODE;
NODE getnode()
{
NODE x;
x=(NODE)malloc(sizeof(struct node));
if(x==NULL)
 {
  printf("memory full\n");
  exit(0);
 }
 return x;
}

void freenode(NODE x)
{
free(x);
}

NODE insert_front(NODE first,int item)
{
NODE temp;
temp=getnode();
temp->info=item;
temp->link=NULL;
```

```
if(first==NULL)
return temp;
temp->link=first;
first=temp;
return first;
}

NODE delete_front(NODE first)
{
NODE temp;
if(first==NULL)
{
printf("list is empty cannot delete\n");
return first;
}
temp=first;
temp=temp->link;
printf("item deleted at front-end is=%d\n",first->info);
free(first);
return temp;
}

NODE insert_rear(NODE first,int item)
{
NODE temp,cur;
temp=getnode();
temp->info=item;
temp->link=NULL;
if(first==NULL)
 return temp;
cur=first;
while(cur->link!=NULL)
 cur=cur->link;
cur->link=temp;
return first;
}

NODE delete_rear(NODE first)
{
NODE cur,prev;
if(first==NULL)
{
printf("list is empty cannot delete\n");
return first;
}
if(first->link==NULL)
```

```
{
printf("item deleted is %d\n",first->info);
free(first);
return NULL;
}
prev=NULL;
cur=first;
while(cur->link!=NULL)
{
prev=cur;
cur=cur->link;
}
printf("item deleted at rear-end is %d",cur->info);
free(cur);
prev->link=NULL;
return first;
}

NODE delete_info(int key,NODE first)
{
NODE prev,cur;
if(first==NULL)
{
printf("list is empty\n");
return NULL;
}
if(key==first->info)
{
cur=first;
first=first->link;
freenode(cur);
return first;
}
prev=NULL;
cur=first;
while(cur!=NULL)
{
if(key==cur->info)break;
prev=cur;
cur=cur->link;
}
if(cur==NULL)
{
printf("search is unsuccessfull\n");
return first;
}
```

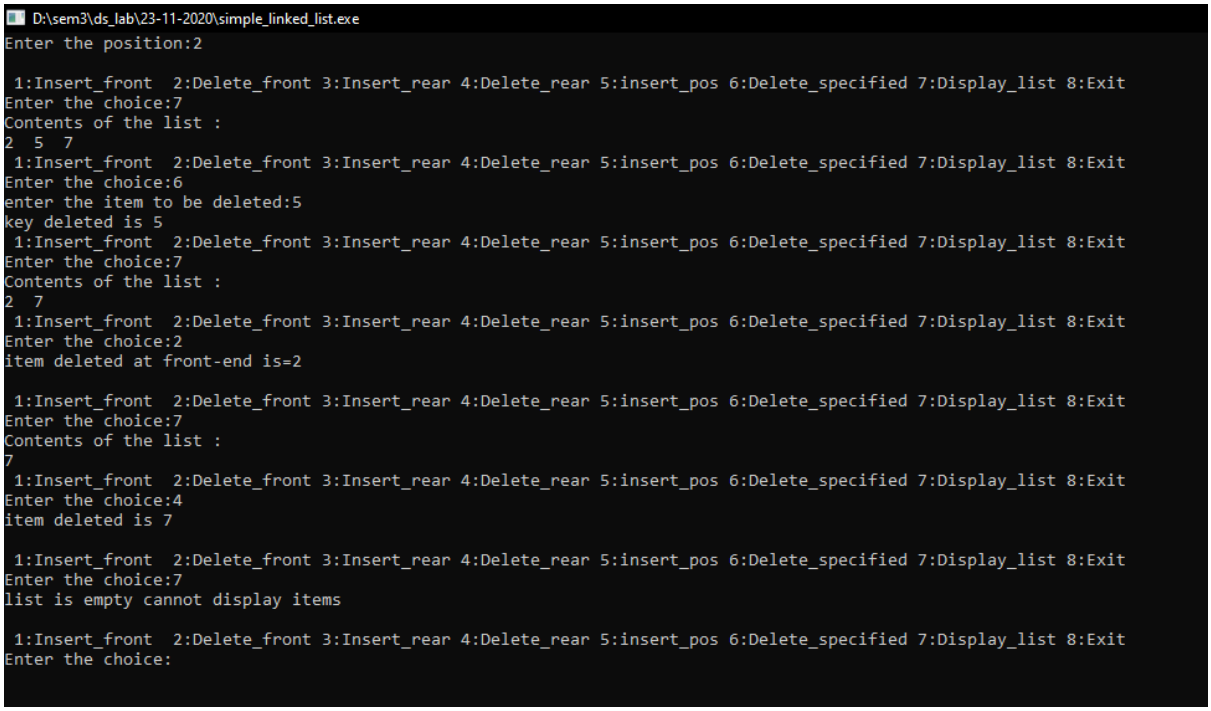
```
prev->link=cur->link;
printf("key deleted is %d",cur->info);
freenode(cur);
return first;
}

NODE insert_pos( int item, int pos, NODE first)
{
NODE temp;
NODE prev,cur;
int count;
temp=getnode();
temp->info=item;
temp->link=NULL;
if(first==NULL && pos==1)
{
return temp;
}
if(first==NULL)
{
printf("invalid position\n");
return first;
}
if(pos==1)
{
temp->link=first;
return temp;
}
count=1;
prev=NULL;
cur=first;
while(cur!=NULL && count!=pos)
{
prev=cur;
cur=cur->link;
count++;
}
if(count==pos)
{
prev->link=temp;
temp->link=cur;
return first;
}
 printf("invalid position\n");
return first;
```

```
}


void display(NODE first)
{
 NODE temp;
 if(first==NULL)
 printf("list is empty cannot display items\n");
 else
 {
       printf("Contents of the list : \n");
      for(temp=first;temp!=NULL;temp=temp->link)
      {
            printf("%d  ",temp->info);
      }
 }
}
int main()
{
int item,choice,key,pos;
NODE first=NULL;
for(;;)
{
printf("\n 1:Insert_front  2:Delete_front 3:Insert_rear
4:Delete_rear 5:insert_pos 6:Delete_specified 7:Display_list
8:Exit\n");
printf("Enter the choice:");
scanf("%d",&choice);
switch(choice)
 {
  case 1:printf("Enter the item at front-end:");
     scanf("%d",&item);
     first=insert_front(first,item);
     break;
  case 2:first=delete_front(first);
     break;
  case 3:printf("Enter the item at rear-end: ");
     scanf("%d",&item);
     first=insert_rear(first,item);
     break;
  case 4:first=delete_rear(first);
     break;
  case 5:printf("Enter the item to be inserted:");
     scanf("%d",&item);
     printf("Enter the position:");
     scanf("%d",&pos);
```

```
      insert_pos( item, pos, first);
      break;
   case 6:printf("enter the item to be deleted:");
      scanf("%d",&key);
      first=delete_info(key,first);
      break;
   case 7:display(first);
      break;
  default:exit(0);
      break;
  }
}
getch();
return 0;
}
```

Output:

```
D:\sem3\ds_lab\23-11-2020\simple_linked_list.exe

 1:Insert_front  2:Delete_front 3:Insert_rear 4:Delete_rear 5:insert_pos 6:Delete_specified 7:Display_list 8:Exit
Enter the choice:2
list is empty cannot delete

 1:Insert_front  2:Delete_front 3:Insert_rear 4:Delete_rear 5:insert_pos 6:Delete_specified 7:Display_list 8:Exit
Enter the choice:4
list is empty cannot delete

 1:Insert_front  2:Delete_front 3:Insert_rear 4:Delete_rear 5:insert_pos 6:Delete_specified 7:Display_list 8:Exit
Enter the choice:6
enter the item to be deleted:6
list is empty

 1:Insert_front  2:Delete_front 3:Insert_rear 4:Delete_rear 5:insert_pos 6:Delete_specified 7:Display_list 8:Exit
Enter the choice:1
Enter the item at front-end:2

 1:Insert_front  2:Delete_front 3:Insert_rear 4:Delete_rear 5:insert_pos 6:Delete_specified 7:Display_list 8:Exit
Enter the choice:3
Enter the item at rear-end: 7

 1:Insert_front  2:Delete_front 3:Insert_rear 4:Delete_rear 5:insert_pos 6:Delete_specified 7:Display_list 8:Exit
Enter the choice:7
Contents of the list :
2  7
 1:Insert_front  2:Delete_front 3:Insert_rear 4:Delete_rear 5:insert_pos 6:Delete_specified 7:Display_list 8:Exit
Enter the choice:5
Enter the item to be inserted:5
Enter the position:2

 1:Insert_front  2:Delete_front 3:Insert_rear 4:Delete_rear 5:insert_pos 6:Delete_specified 7:Display_list 8:Exit
Enter the choice:7
Contents of the list :
2  5  7
```


D:\sem3\ds_lab\23-11-2020\simple_linked_list.exe
Enter the position:2
1:Insert_front 2:Delete_front 3:Insert_rear 4:Delete_rear 5:insert_pos 6:Delete_specified 7:Display_list 8:Exit
Enter the choice:7
Contents of the list :
2 5 7
1:Insert_front 2:Delete_front 3:Insert_rear 4:Delete_rear 5:insert_pos 6:Delete_specified 7:Display_list 8:Exit
Enter the choice:6
enter the item to be deleted:5
key deleted is 5
1:Insert_front 2:Delete_front 3:Insert_rear 4:Delete_rear 5:insert_pos 6:Delete_specified 7:Display_list 8:Exit
Enter the choice:7
Contents of the list :
2 7
1:Insert_front 2:Delete_front 3:Insert_rear 4:Delete_rear 5:insert_pos 6:Delete_specified 7:Display_list 8:Exit
Enter the choice:2
item deleted at front-end is=2
1:Insert_front 2:Delete_front 3:Insert_rear 4:Delete_rear 5:insert_pos 6:Delete_specified 7:Display_list 8:Exit
Enter the choice:7
Contents of the list :
7
1:Insert_front 2:Delete_front 3:Insert_rear 4:Delete_rear 5:insert_pos 6:Delete_specified 7:Display_list 8:Exit
Enter the choice:4
item deleted is 7
1:Insert_front 2:Delete_front 3:Insert_rear 4:Delete_rear 5:insert_pos 6:Delete_specified 7:Display_list 8:Exit
Enter the choice:7
list is empty cannot display items
1:Insert_front 2:Delete_front 3:Insert_rear 4:Delete_rear 5:insert_pos 6:Delete_specified 7:Display_list 8:Exit
Enter the choice:

**********