Lab5:--

WAP to Implement Singly Linked List with following operations

a) 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.

Program:-

```
#include<stdio.h>
#include<stdlib.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("mem 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!Can't delete an item'\n");
return first;
}
temp=first;
temp=temp->link;
printf("The 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 insert_pos(int item,int pos,NODE first)
{
NODE temp,cur,prev;
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;
first=temp;
return temp;
}
count=1;
prev=NULL;
cur=first;
```

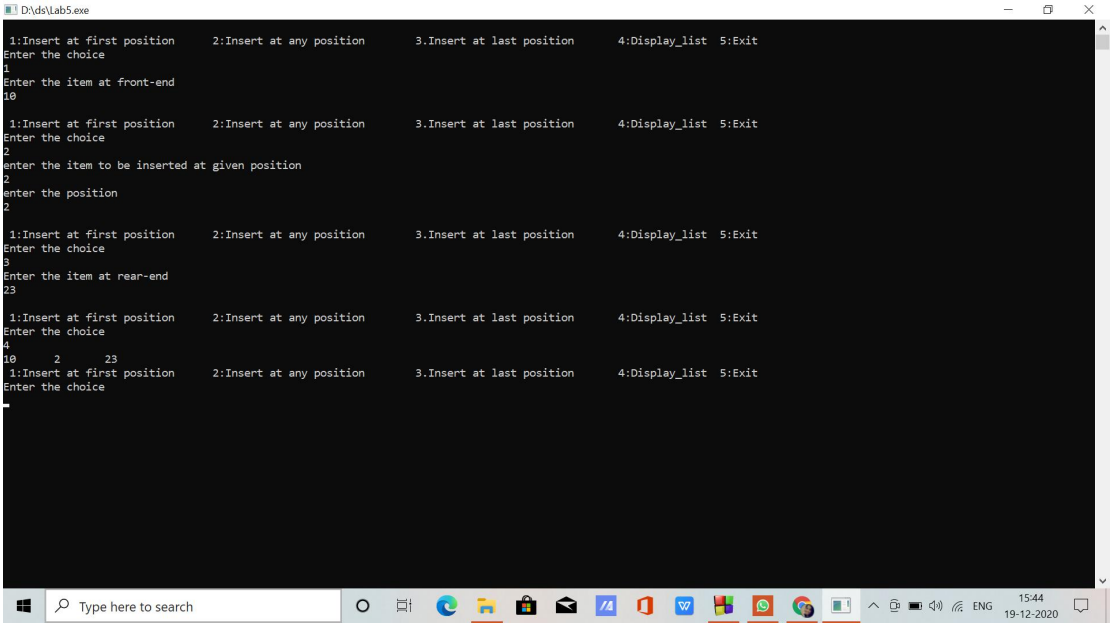
```
while(cur!=NULL&&count!=pos)
{
prev=cur;
cur=cur->link;
count++;
}
if(count==pos)
{
prev->link=temp;
temp->link=cur;
return first;
}
printf("invalid position\n");
return first;
}


void display(NODE first)
{
 NODE temp;
 if(first==NULL)
 printf("List empty cannot display items\n");
 for(temp=first;temp!=NULL;temp=temp->link)
  {
  printf("%d\t",temp->info);
  }
}

int main()
{
int item,choice,pos;
NODE first=NULL;
for(;;)
{
printf("\n 1:Insert at first position\t 2:Insert at any position\t 3.Insert at last position\t
4:Display_list\t 5:Exit\n");
printf("Enter the choice\n");
scanf("%d",&choice);
switch(choice)
 {
  case 1:printf("Enter the item at front-end\n");
         scanf("%d",&item);
         first=insert_front(first,item);
         break;
```

```
 case 2:printf("enter the item to be inserted at given position\n");
        scanf("%d",&item);
        printf("enter the position\n");
        scanf("%d",&pos);
        first=insert_pos(item,pos,first);
        break;
 case 3:printf("Enter the item at rear-end\n");
        scanf("%d",&item);
        first=insert_rear(first,item);
        break;
 case 4:display(first);
        break;
 default:exit(0);
        break;
 }
}
}
```

Output screenshot:-



Written Screenshot:-

Lab-6

```
#include <stdio.h>
#include <stdlib.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("mem 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 IF (NODE second, int item)
{
NODE temp;
temp = getnode();
temp -> info = item;
temp -> link = NULL
if (second == NULL)
return temp;
temp -> link = second;
second = temp;
return second;
}
NODE insert_pos (int item, int pos, NODE fi
{
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)
{
pf("invalid pos\n");
return first;
}
if (pos == 1)
{
temp -> link = first;
return temp;
}
if (count == pos)
{
prev -> link = temp;
temp -> link = cur;
return first;
}
void display (NODE first)
{
NODE temp;
if (first == NULL)
printf("list empty cannot display items\n")
for (temp = first; temp != NULL; temp = temp -> li
pf("%d \n", temp -> info);
}
}
int main()
{
int item, choice, pos, element;
NODE first = NULL;
NODE second = NULL;
for(;;)
{
pf("1n1 insert @ first pos \n2 insert @ last
\n3 - insert @ specific pos \n4 display \n5 e
pf("enter choice");
sf("%d", &choice);
switch(choice)
{
case 1: pf("enter the item @ front end \n");
sf("%d", &item);
first = insert_front(first);
break;
case 2: printf("enter the item @ rearend")
sf("%d", &item);
first = insert_rear(first, item);
break;
```

DATE: / /

```
case 3:
pf ("enter the position to insert");
sf ("%d", &pos);
pf ("enter the item to insert");
sf ("%d", &item);
first = insert_pos(item, pos, first);
break;
case 4: display(first);
 break;
default : exit(0);
    break;
  }
}
}
```