1BM19CS079

LIKITHA B

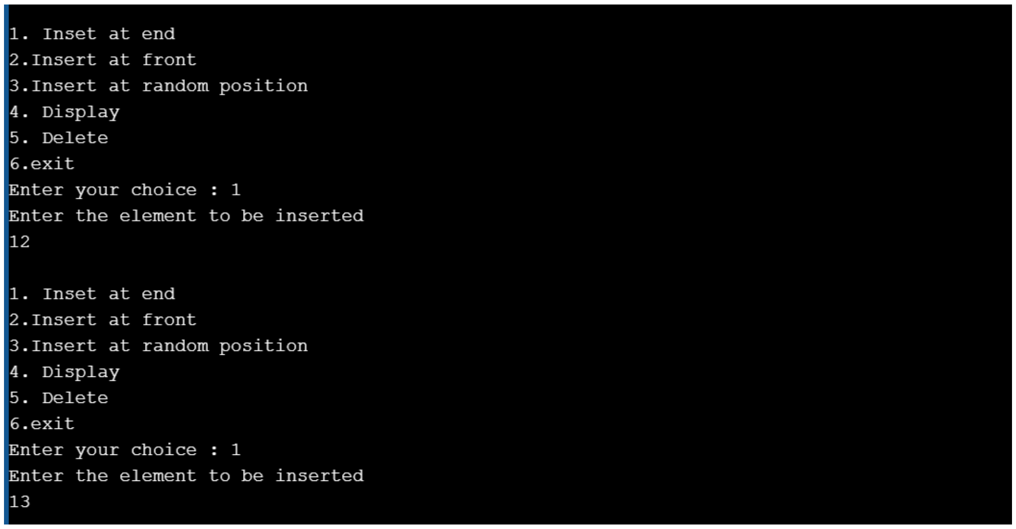
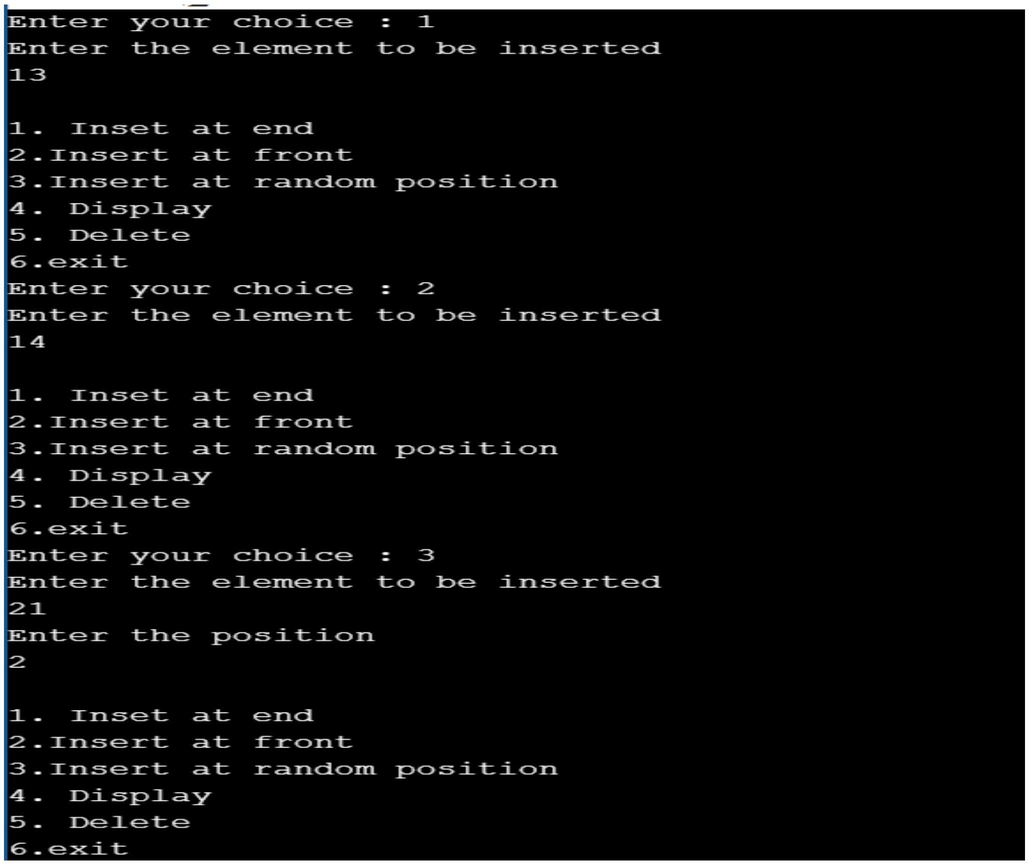
PROGRAM-1 AND 2

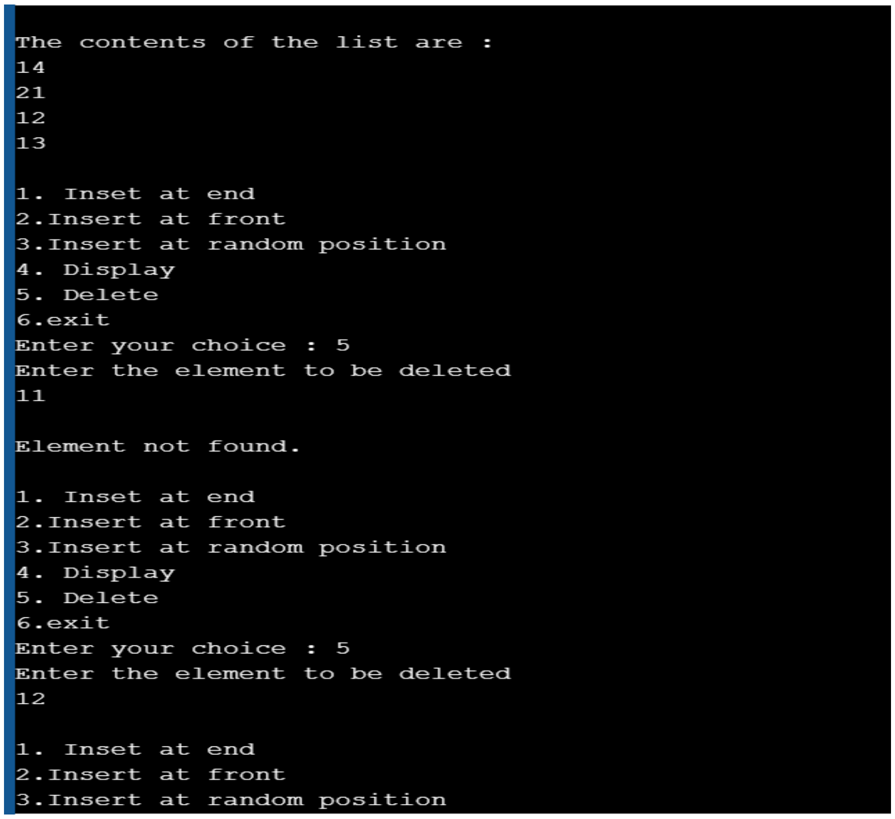
```
#include<stdio.h>


struct node

{

   int data;

   struct node *next;

};


struct node *head=NULL;


int length=0;


void insertend(int ele)

{

   struct node *newnode,*temp;

   newnode=(struct node*)malloc(sizeof(struct node));

   newnode->data=ele;

   newnode->next=NULL;

   if(head==NULL)

   {

      head=newnode;
}

      length=1;

   }

   else

   {

      temp=(struct node*)malloc(sizeof(struct node));

      temp=head;

      while(temp->next!=NULL)

      {

          temp=temp->next;

      }

      temp->next=newnode;

      length++;

   }


}


void insertfront(int ele)

{

   struct node *temp;

   temp=(struct node*)malloc(sizeof(struct node));

   temp->data=ele;

   temp->next=head;

   head=temp;

   length++;

}
```

```
void insertrandom(int ele,int pos)
{
   if(pos==1)
      insertfront(ele);
   else if(pos>=length)
      insertend(ele);
   else
   {
      struct node *inst;
      inst=(struct node*)malloc(sizeof(struct node));
      struct node *temp;
      temp=(struct node*)malloc(sizeof(struct node));
      temp=head;
      for(int i=1;i<pos-1;i++)
      {
           temp=temp->next;
      }
      inst->data=ele;
      inst->next=temp->next;
      temp->next=inst;
      length++;


   }
```

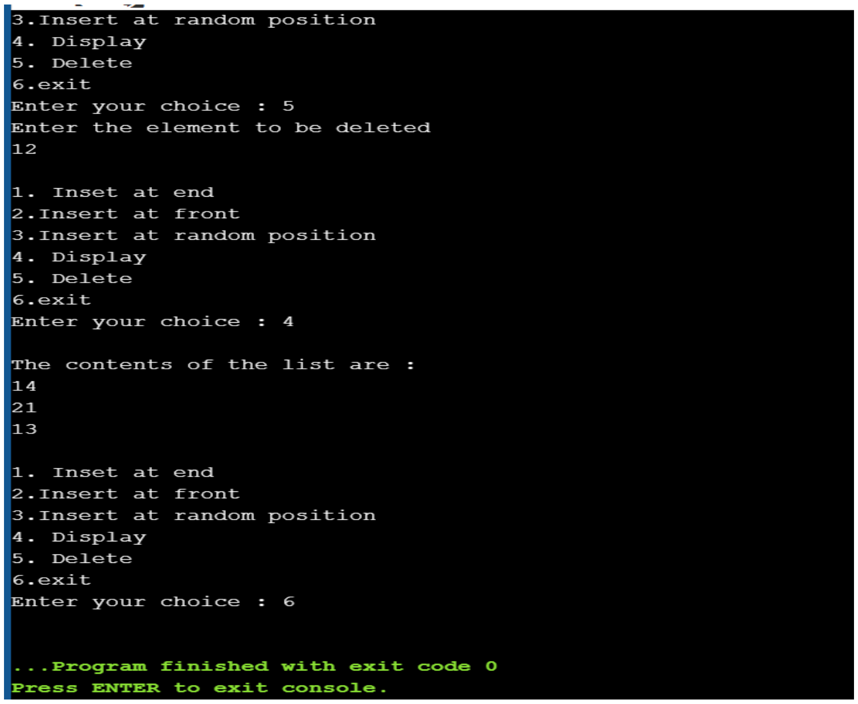
```
}


void deleteele(int ele)

{

    struct node *temp,*del;

    temp=(struct node*)malloc(sizeof(struct node));

    del=(struct node*)malloc(sizeof(struct node));

    del=NULL;

    if(head->data==ele)

    {

       del=head;

       head=head->next;

       del->next=NULL;

    }

    else

    {

       temp=head;

       while(temp->next!=NULL)

       {

         if(temp->next->data==ele)

         {


              del=temp->next;

              temp->next=del->next;

              del->next=NULL;
```

```
                length--;
                break;
            }
            else
            {
                temp=temp->next;
            }

        }
    }
    if(del==NULL)
    {
        printf("\nElement not found.\n");
    }
}

void display()
{
    struct node *temp;
    temp=(struct node*)malloc(sizeof(struct node));
    temp=head;
    if(temp==NULL)
    {
        printf("\n List is empty \n");
    }
    else
    {
        printf("\nThe contents of the list are :\n");
        while(temp!=NULL)
        {
            printf("%d\n",temp->data);
            temp=temp->next;
        }
    }

}

int main()
{
    int choice,ele,pos;
    char ch;
    do
    {
    printf("\n1. Inset at end \n2.Insert at front \n3.Insert at random position \n4. Display  \n5. Delete 
\n6.exit");
    printf("\nEnter your choice : ");
    scanf("%d",&choice);
    switch(choice)
    {
        case 1: printf("Enter the element to be inserted\n");
                scanf("%d",&ele);
                insertend(ele);

                break;

        case 2: printf("Enter the element to be inserted\n");

                scanf("%d",&ele);

                insertfront(ele);

                break;

        case 3: printf("Enter the element to be inserted\n");

                scanf("%d",&ele);

                printf("Enter the position \n");

                scanf("%d",&pos);

                insertrandom(ele,pos);

                break;

        case 4: display();

                break;

        case 5: printf("Enter the element to be deleted\n");

                scanf("%d",&ele);

                deleteele(ele);

                break;

    }

    }while(choice!=6);

    return 0;

}
```


```
1. Inset at end
2.Insert at front
3.Insert at random position
4. Display
5. Delete
6.exit
Enter your choice : 1
Enter the element to be inserted
12

1. Inset at end
2.Insert at front
3.Insert at random position
4. Display
5. Delete
6.exit
Enter your choice : 1
Enter the element to be inserted
13
```

```
Enter your choice : 1
Enter the element to be inserted
13

1. Inset at end
2.Insert at front
3.Insert at random position
4. Display
5. Delete
6.exit
Enter your choice : 2
Enter the element to be inserted
14

1. Inset at end
2.Insert at front
3.Insert at random position
4. Display
5. Delete
6.exit
Enter your choice : 3
Enter the element to be inserted
21
Enter the position
2

1. Inset at end
2.Insert at front
3.Insert at random position
4. Display
5. Delete
6.exit
```



```
The contents of the list are :
14
21
12
13

1. Inset at end
2.Insert at front
3.Insert at random position
4. Display
5. Delete
6.exit
Enter your choice : 5
Enter the element to be deleted
11

Element not found.

1. Inset at end
2.Insert at front
3.Insert at random position
4. Display
5. Delete
6.exit
Enter your choice : 5
Enter the element to be deleted
12

1. Inset at end
2.Insert at front
3.Insert at random position
```



```
3.Insert at random position
4. Display
5. Delete
6.exit
Enter your choice : 5
Enter the element to be deleted
12

1. Inset at end
2.Insert at front
3.Insert at random position
4. Display
5. Delete
6.exit
Enter your choice : 4

The contents of the list are :
14
21
13

1. Inset at end
2.Insert at front
3.Insert at random position
4. Display
5. Delete
6.exit
Enter your choice : 6


...Program finished with exit code 0
Press ENTER to exit console.
```

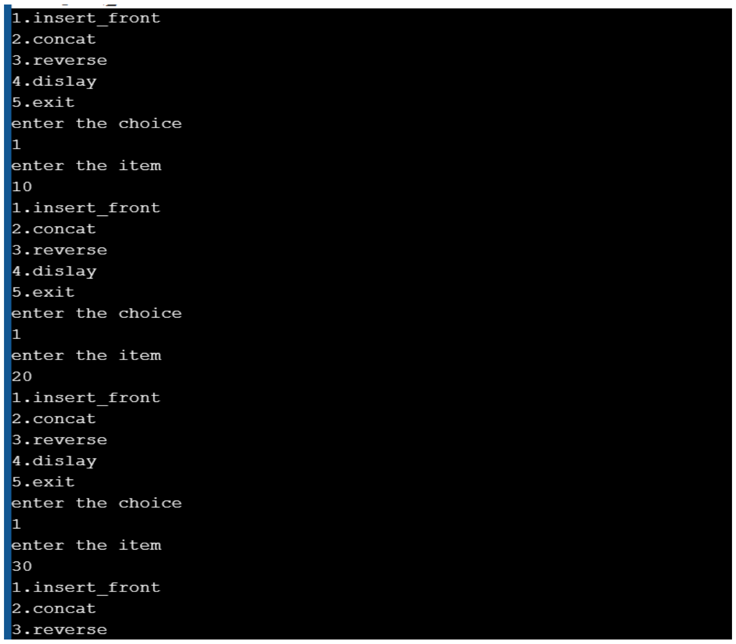
PROGRAM-2(SORTING,CONCATENATION,REVERSE)

```
#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;

}

NODE insert_rear(NODE first,int item)

{

NODE temp,cur;

temp=getnode();

temp->info=item;

temp->link=NULL;
```

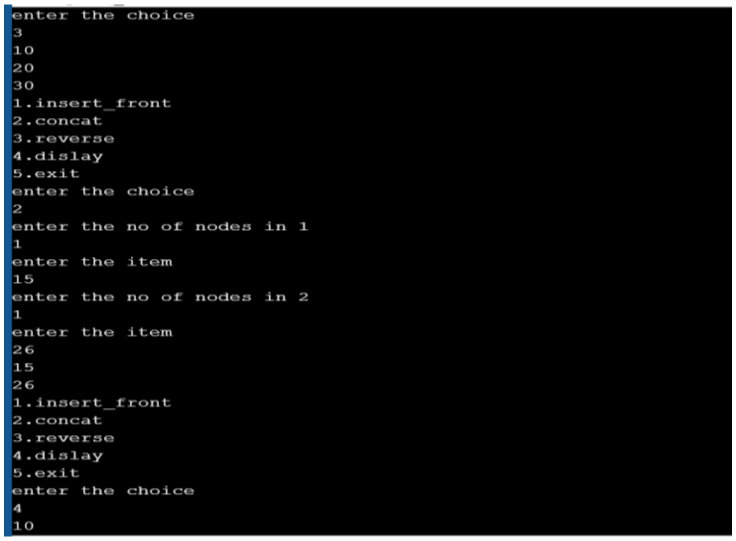
```
if(first==NULL)

return temp;

cur=first;

while(cur->link!=NULL)

cur=cur->link;

cur->link=temp;

return first;

}

void display(NODE first)

{

NODE temp;

if(first==NULL)

printf("list empty");

for(temp=first;temp!=NULL;temp=temp->link)

{

printf("%d\n",temp->info);

}

}

NODE concat(NODE first,NODE second)

{

NODE cur;

if(first==NULL)
```

```
return second;

if(second==NULL)

return first;

cur=first;

while(cur->link!=NULL)

cur=cur->link;

cur->link=second;

return first;

}

NODE reverse(NODE first)

{

NODE cur,temp;

cur=NULL;

while(first!=NULL)

{

temp=first;

first=first->link;

temp->link=cur;

cur=temp;

}

return cur;

}

int main()

{

int item,choice,pos,i,n;
```

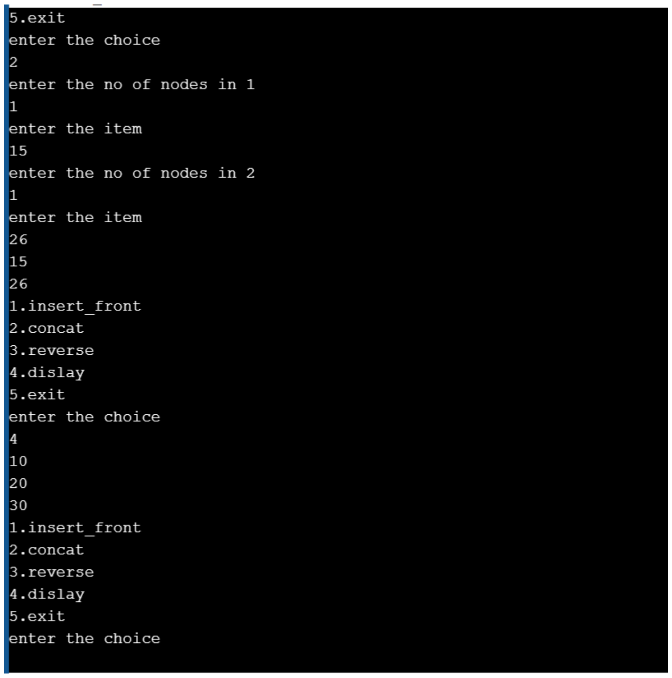
```
NODE first=NULL,a,b;


for(;;)

{

printf("1.insert_front\n2.concat\n3.reverse\n4.dislay\n5.exit\n");

printf("enter the choice\n");

scanf("%d",&choice);

switch(choice)

{

case 1:printf("enter the item\n");

scanf("%d",&item);

first=insert_rear(first,item);

break;


case 2:printf("enter the no of nodes in 1\n");


scanf("%d",&n);

a=NULL;

for(i=0;i<n;i++)

{

printf("enter the item\n");

scanf("%d",&item);

a=insert_rear(a,item);

}
printf("enter the no of nodes in 2\n");

scanf("%d",&n);

b=NULL;

for(i=0;i<n;i++)

{

printf("enter the item\n");

scanf("%d",&item);

b=insert_rear(b,item);

}

a=concat(a,b);

display(a);

break;

case 3:first=reverse(first);

display(first);

break;

case 4:display(first);

break;

default:exit(0);

}

}


}
```



```
1.insert_front
2.concat
3.reverse
4.dislay
5.exit
enter the choice
1
enter the item
10
1.insert_front
2.concat
3.reverse
4.dislay
5.exit
enter the choice
1
enter the item
20
1.insert_front
2.concat
3.reverse
4.dislay
5.exit
enter the choice
1
enter the item
30
1.insert_front
2.concat
3.reverse
```



```
enter the choice
3
10
20
30
1.insert_front
2.concat
3.reverse
4.dislay
5.exit
enter the choice
2
enter the no of nodes in 1
1
enter the item
15
enter the no of nodes in 2
1
enter the item
26
15
26
1.insert_front
2.concat
3.reverse
4.dislay
5.exit
enter the choice
4
10
```


```
5.exit
enter the choice
2
enter the no of nodes in 1
1
enter the item
15
enter the no of nodes in 2
1
enter the item
26
15
26
1.insert_front
2.concat
3.reverse
4.dislay
5.exit
enter the choice
4
10
20
30
1.insert_front
2.concat
3.reverse
4.dislay
5.exit
enter the choice
```