DS LAB-PROG 6,7-SINGLY LINKED LIST

Program and output

Mallika Prasad

1BM19CS081

25.11.2020

Program 6-

[SLL including inserting at any position and deleting specified element]

```
#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 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("iten 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 NULL;
}
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)
{
```

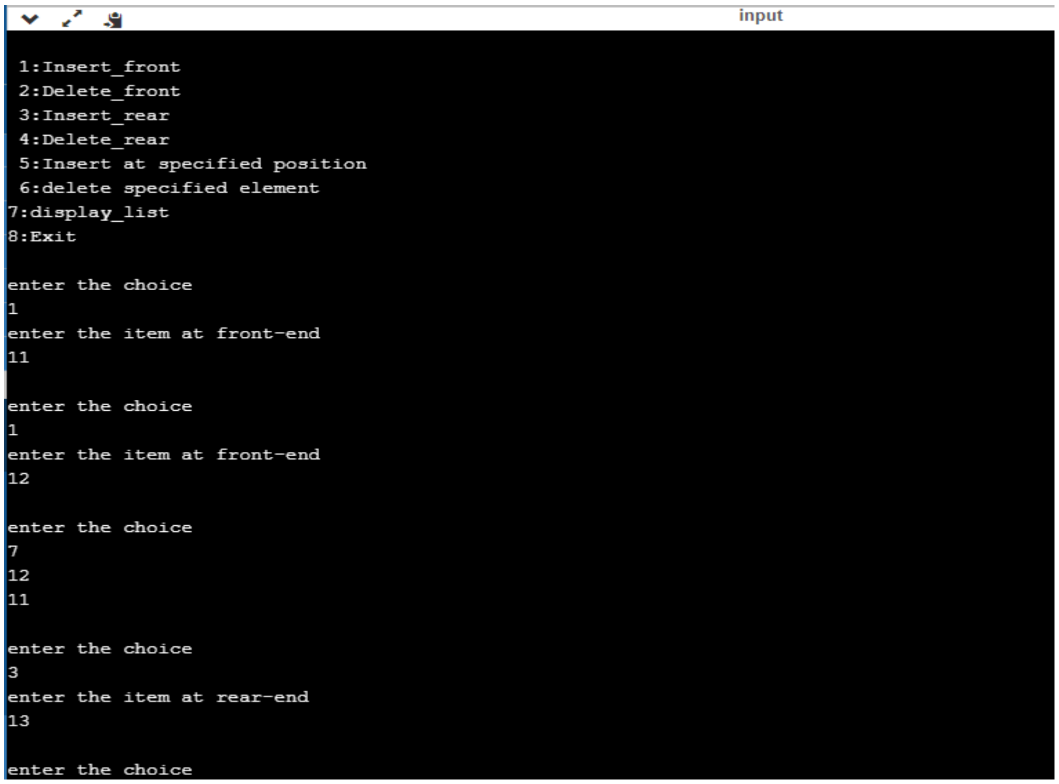
```
                prev->link=temp;
                temp->link=cur;
                return first;
        }
        printf("Invalid position\n");
        return first;
}

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

```
{
if(item==cur->info)break;
prev=cur;
cur=cur->link;
}
if(cur==NULL)
{
printf("search is unsuccessfull\n");
return first;
}
prev->link=cur->link;
printf("item deleted is %d",cur->info);
freenode(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->link)
  {
  printf("%d\n",temp->info);
  }
```

```
}


int main()

{

int item,choice,pos;

NODE first=NULL;

printf("\n 1:Insert_front\n 2:Delete_front\n 3:Insert_rear\n 4:Delete_rear\n 5:Insert at
specified position\n 6:delete specified element \n7:display_list\n8:Exit\n");

do

{

printf("\nenter 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:first=delete_front(first);

             break;

  case 3:printf("enter the item at rear-end\n");

             scanf("%d",&item);

             first=insert_rear(first,item);

             break;

  case 4:first=delete_rear(first);break;

  case 5:printf("Enter the item and the position:\n");

                                        scanf("%d%d",&item,&pos);
```

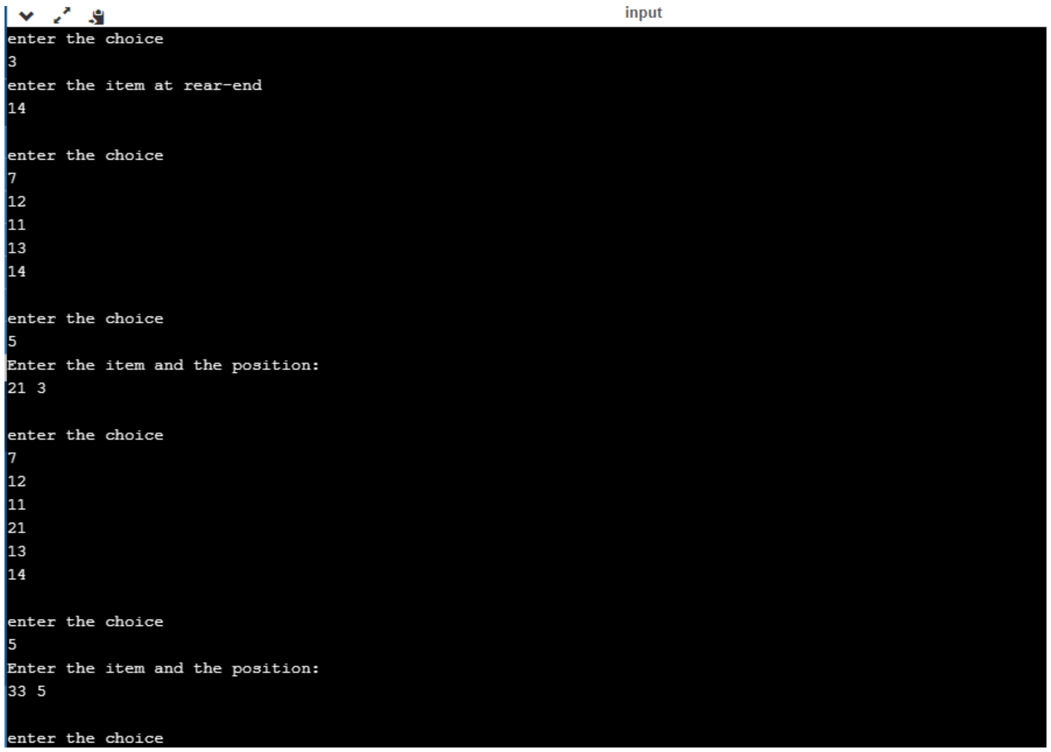
```
                                    first=insert_pos(item,pos,first);
                                    break;
 case 6:printf("enter the element to be deleted\n");
                  scanf("%d",&item);
                  first=delete_info(item,first);
                  break;
  case 7:display(first);
          break;
          case 8:break;
 default:break;
 }
}while(choice!=8);
return 0;
}
```

Output-



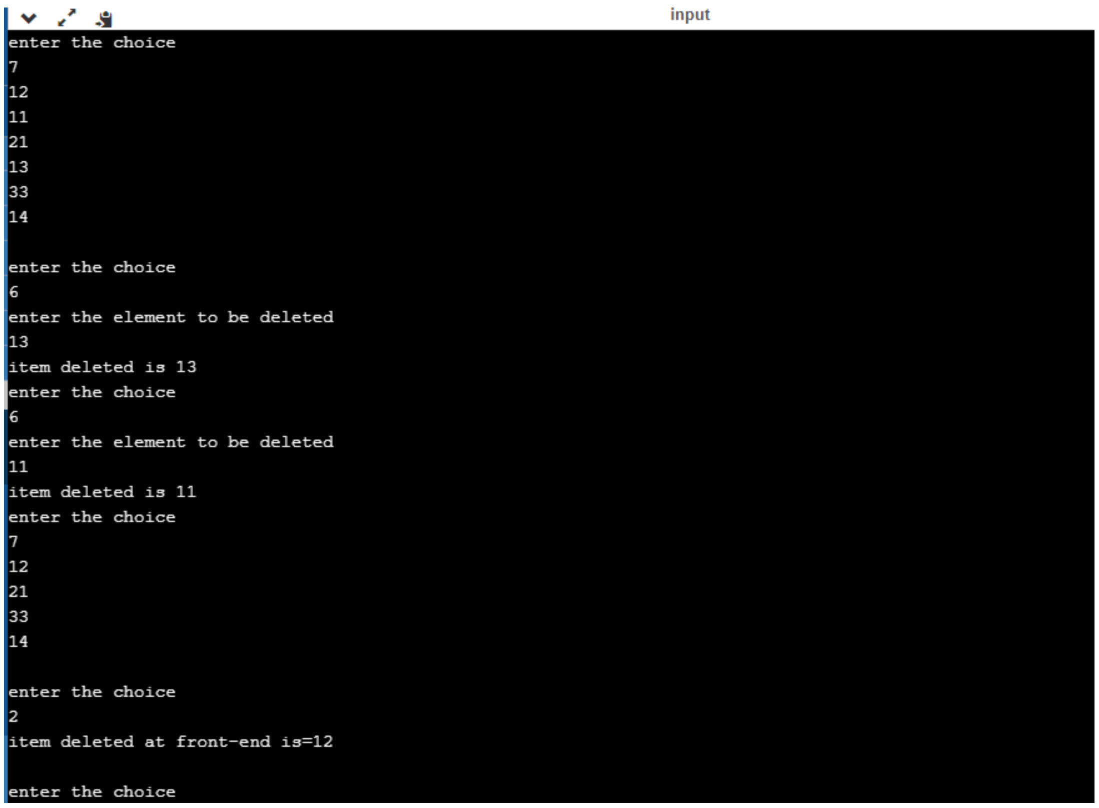
input

```
 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:Insert at specified position
 6:delete specified element
7:display_list
8:Exit

enter the choice
1
enter the item at front-end
11

enter the choice
1
enter the item at front-end
12

enter the choice
7
12
11

enter the choice
3
enter the item at rear-end
13

enter the choice
```

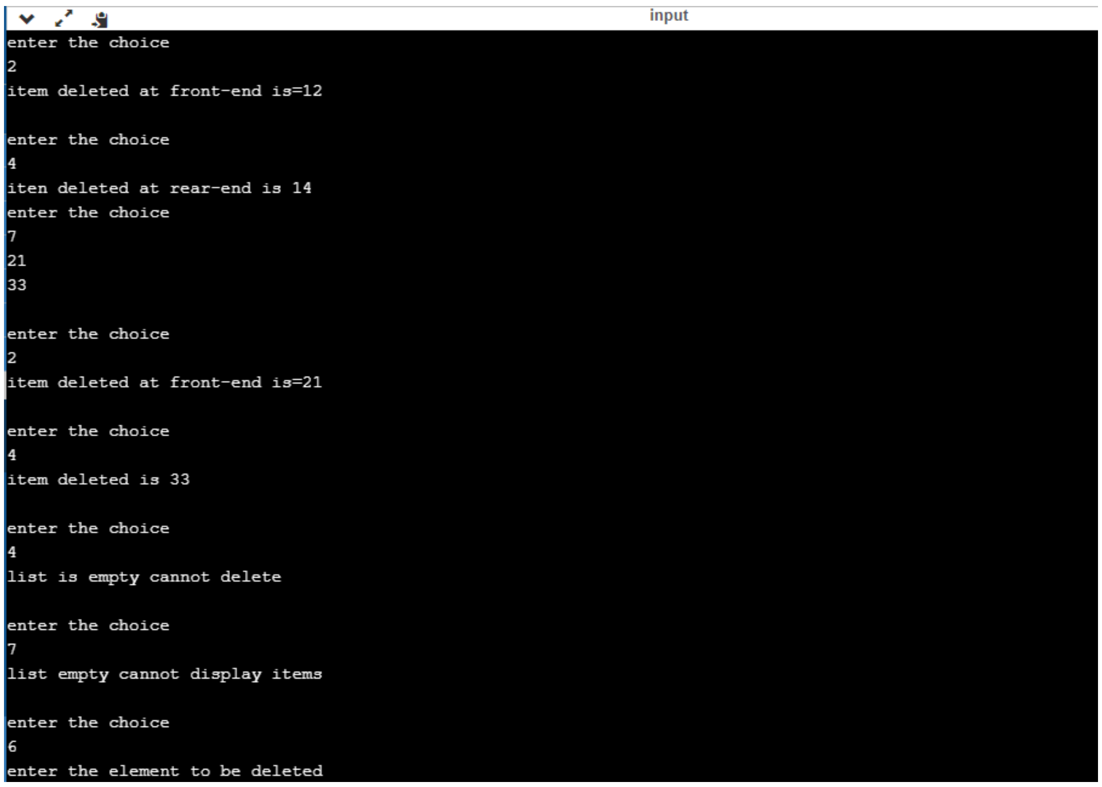


input

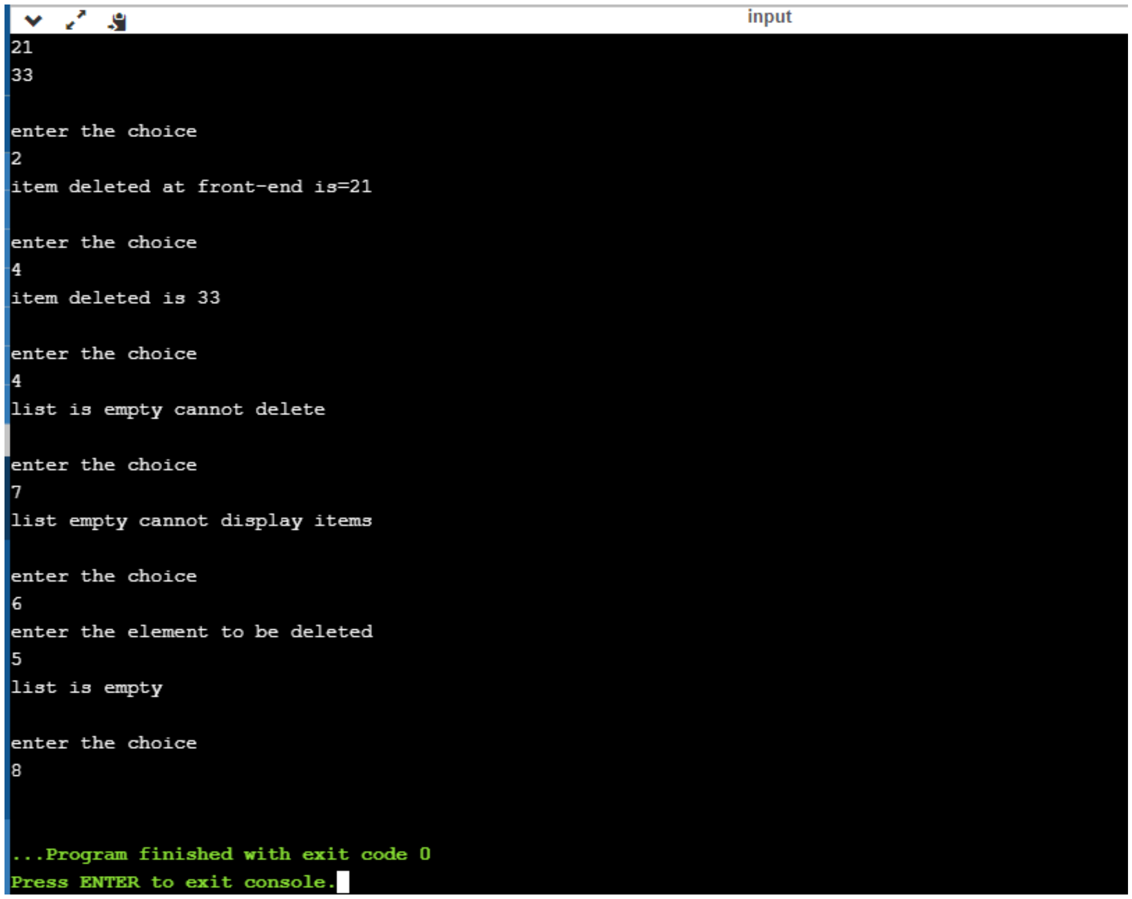
```
enter the choice
3
enter the item at rear-end
14

enter the choice
7
12
11
13
14

enter the choice
5
Enter the item and the position:
21 3

enter the choice
7
12
11
21
13
14

enter the choice
5
Enter the item and the position:
33 5

enter the choice
```


input

```
enter the choice
7
12
11
21
13
33
14

enter the choice
6
enter the element to be deleted
13
item deleted is 13
enter the choice
6
enter the element to be deleted
11
item deleted is 11
enter the choice
7
12
21
33
14

enter the choice
2
item deleted at front-end is=12

enter the choice
```


input

```
enter the choice
2
item deleted at front-end is=12

enter the choice
4
iten deleted at rear-end is 14
enter the choice
7
21
33

enter the choice
2
item deleted at front-end is=21

enter the choice
4
item deleted is 33

enter the choice
4
list is empty cannot delete

enter the choice
7
list empty cannot display items

enter the choice
6
enter the element to be deleted
```



```
21
33

enter the choice
2
item deleted at front-end is=21

enter the choice
4
item deleted is 33

enter the choice
4
list is empty cannot delete

enter the choice
7
list empty cannot display items

enter the choice
6
enter the element to be deleted
5
list is empty

enter the choice
8


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

Program 7-

[a) Sort the linked list. b) Reverse the linked list. c) Concatenation of two linked lists]

```
#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 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("iten deleted at rear-end is %d",cur->info);

free(cur);
```

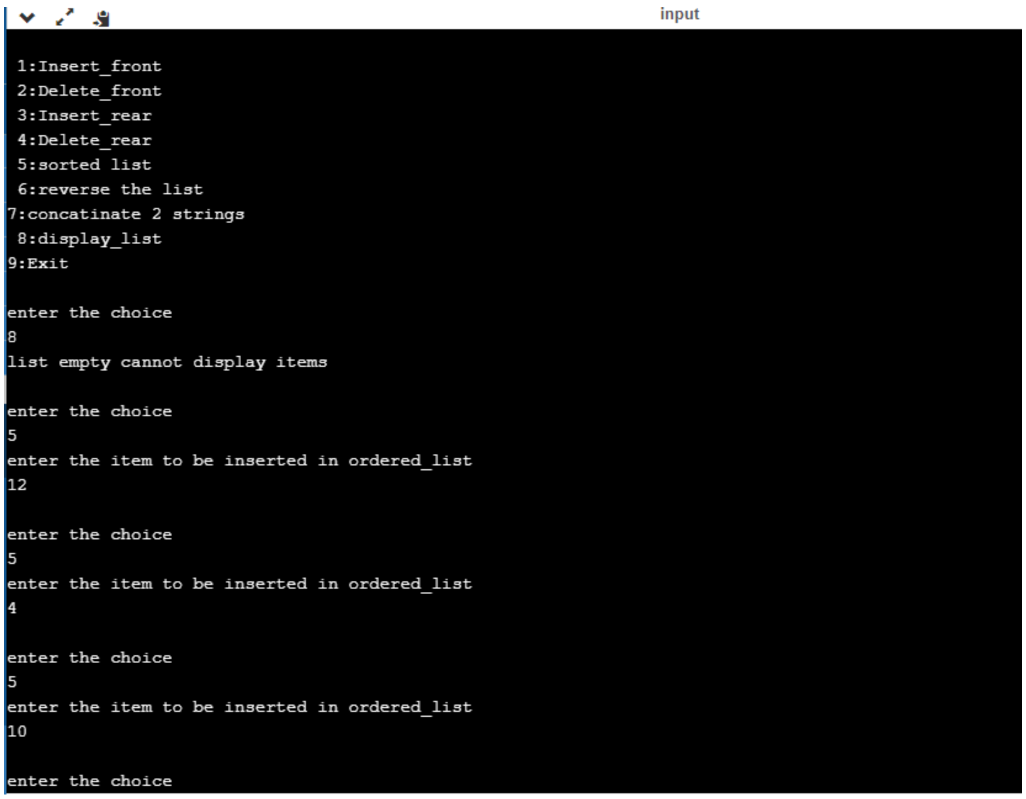
```
prev->link=NULL;

return first;

}


NODE order_list(int item,NODE first)

{

NODE temp,prev,cur;

temp=getnode();

temp->info=item;

temp->link=NULL;

if(first==NULL) return temp;

if(item<first->info)

{

temp->link=first;

return temp;

}

prev=NULL;

cur=first;

while(cur!=NULL&&item>cur->info)

{

prev=cur;

cur=cur->link;

}

prev->link=temp;

temp->link=cur;

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;
}

NODE concat(NODE first,NODE second)
{
NODE cur;
if(first==NULL)
return second;
if(second==NULL)
return first;
cur=first;
while(cur->link!=NULL)
```

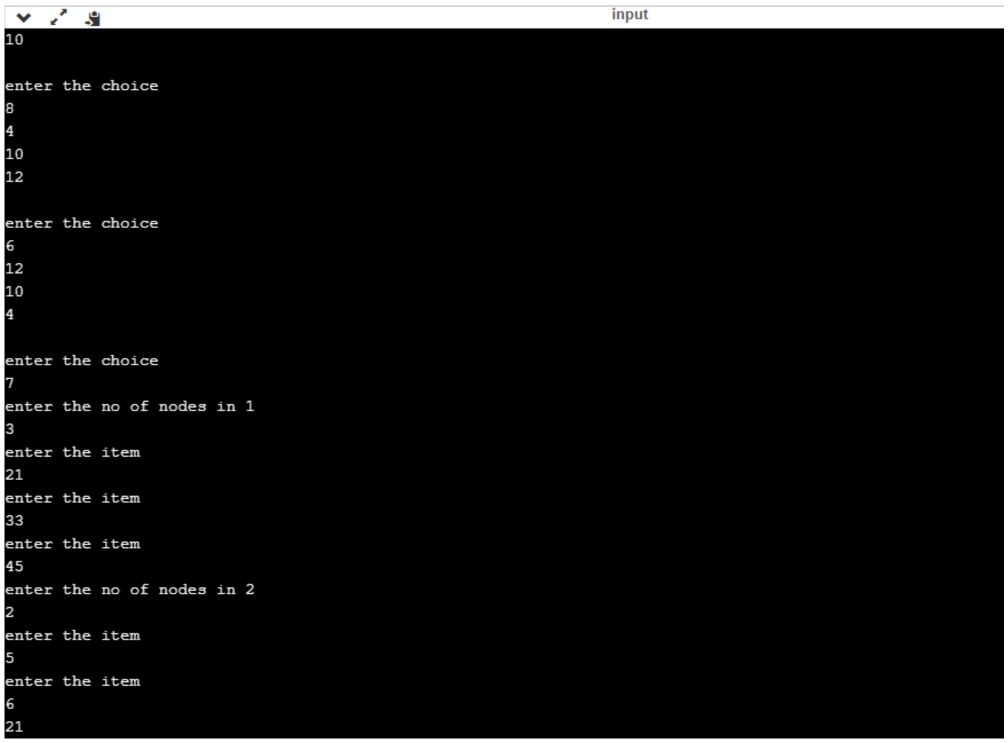
```
cur=cur->link;

cur->link=second;

return first;

}


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\n",temp->info);

  }

}


int main()

{

int item,choice,pos,n;

NODE first=NULL,a,b;


printf("\n 1:Insert_front\n 2:Delete_front\n 3:Insert_rear\n 4:Delete_rear\n 5:sorted list \n
6:reverse the list \n7:concatinate 2 strings\n 8:display_list\n9:Exit\n");

do

{

printf("\nenter 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:first=delete_front(first);
             break;
  case 3:printf("enter the item at rear-end\n");
             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 in ordered_list\n");
        scanf("%d",&item);
        first=order_list(item,first);
        break;
 case 6:first=reverse(first);
         display(first);
         break;
 case 7:printf("enter the no of nodes in 1\n");
         scanf("%d",&n);
         a=NULL;
         for(int 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(int 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 8:display(first);
            break;
            case 9:break;
 default:break;
 }
}while(choice!=9);
return 0;
}
```

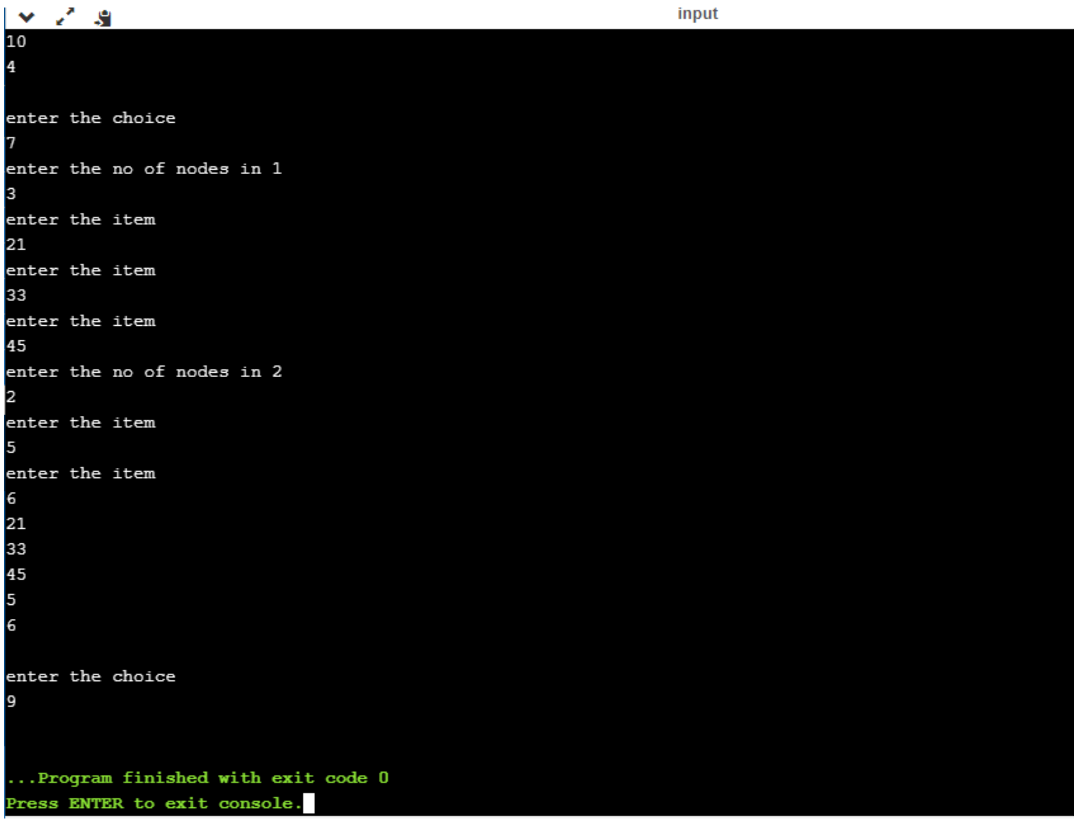
Output-



```
 1:Insert_front
 2:Delete_front
 3:Insert_rear
 4:Delete_rear
 5:sorted list
 6:reverse the list
7:concatinate 2 strings
 8:display_list
9:Exit

enter the choice
8
list empty cannot display items

enter the choice
5
enter the item to be inserted in ordered_list
12

enter the choice
5
enter the item to be inserted in ordered_list
4

enter the choice
5
enter the item to be inserted in ordered_list
10

enter the choice
```



```
10

enter the choice
8
4
10
12

enter the choice
6
12
10
4

enter the choice
7
enter the no of nodes in 1
3
enter the item
21
enter the item
33
enter the item
45
enter the no of nodes in 2
2
enter the item
5
enter the item
6
21
```


input

```
10
4

enter the choice
7
enter the no of nodes in 1
3
enter the item
21
enter the item
33
enter the item
45
enter the no of nodes in 2
2
enter the item
5
enter the item
6
21
33
45
5
6

enter the choice
9


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