19BCE0829
KANDRA KSHEERAJ

CSE2003- Data Structures and Algorithms

LAB ASSIGNMENT-2

Slot: L53 + L54

Faculty: GAYATHRI P Mam

Date: 21-8-2020

19BCE0829 KSHEERAJ KANDRA

QUESTION-1:

1. Menu-driven C program implement circular queue using array. Perform enqueue, dequeue and display operations.

PSEUDO CODE:

```
define max, q[10], front=0, rear=-1
main()
void insert,void delet,void display
read input ch
switch(ch)
     case 1: insert();
     case 2: delet();
     case 3: display();
     case 4: exit(0);

void insert()
if full queue condition (Rear == max and front ==1) or (rear+1
== front)
     print Queue is overflow
else print "Enter element to be insert"
read input x
If (Rear == max and front ==1) or (rear+1 == front)
     Then Print "OVERFLOW"
If rear == MAX
     rear = 1
else rear = rear +1
      Set QUEUE[rear]=ITEM
If FRONT= 0 //queue is initially empty
     Then Set FRONT=1
```

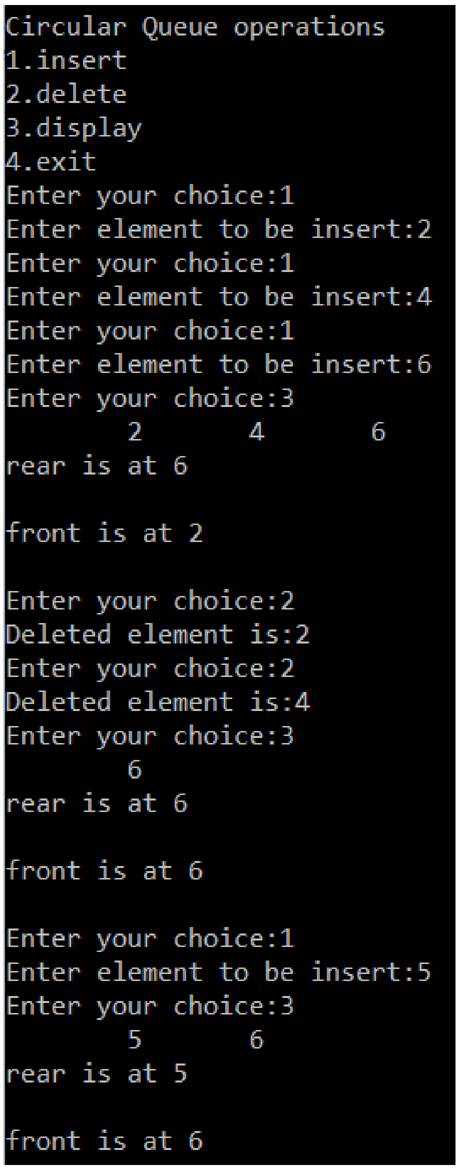
```
void delet()
If FRONT=0, then print "UNDERFLOW"
Else Set ITEM=QUEUE[FRONT]
If FRONT=REAR, then //Queue has only one element]
     Set FRONT=0 and REAR=0
Else if (FRONT = MAX) Then set Front = 1
else Set FRONT=FRONT+1
Print ITEM

void display()
if front=0 and rear=-1
     print "Queue is underflow"
if(front>rear)
     for(i=0;i<=rear;i++)
          print q[i]
     for(j=front;j<=max-1;j++)
          print q[j])
          print rear is at q[rear]
          print front is at q[front]
else{ for(i=front;i<=rear;i++)
     print q[i]
     print rear is at q[rear]
     print front is at q[front]
```

PROGRAM CODE:

```
#include<stdio.h>
#define max 3
int q[10],front=0,rear=-1;
void main(){
int ch;
void insert();
void delet();
void display();
printf("\nCircular Queue operations\n");
printf("1.insert\n2.delete\n3.display\n4.exit\n");
while(1){
printf("Enter your choice:");
scanf("%d",&ch);
```

```
switch(ch){
     case 1: insert();
     break;
     case 2: delet();
     break;
     case 3: display();
     break;
     case 4: exit(0);
     break;
     default:printf("Invalid option\n");}
     }
}
void insert(){
int x;
if((front==0&&rear==max-1)||(front>0&&rear==front-1))
     printf("Queue is overflow\n");
else
     printf("Enter element to be insert:");
scanf("%d",&x);
if(rear==max-1&&front>0){
    rear=0;
    q[rear]=x;
}
else{
if((front==0&&rear==-1)||(rear!=front-1))
    q[++rear]=x;
}}
void delet(){
int a;
if((front==0)&&(rear==-1)){
     printf("Queue is underflow\n");
}
if(front==rear){
    a=q[front];
    rear=-1;
    front=0;
}
else
if(front==max-1){
    a=q[front];
    front=0;
}
else a=q[front++];
    printf("Deleted element is:%d\n",a);
}
void display(){
int i,j;
if(front==0&&rear==-1){
    printf("Queue is underflow\n");
}
if(front>rear){
for(i=0;i<=rear;i++)
    printf("\t%d",q[i]);
for(j=front;j<=max-1;j++)
    printf("\t%d",q[j]);
     printf("\nrear is at %d\n",q[rear]);
     printf("\nfront is at %d\n",q[front]);
}
else {
for(i=front;i<=rear;i++){
     printf("\t%d",q[i]);
     }
    printf("\nrear is at %d\n",q[rear]);
    printf("\nfront is at %d\n",q[front]);
     }
    printf("\n");
}
```

PROGRAM OUTPUT:



```
Circular Queue operations
1.insert
2.delete
3.display
4.exit
Enter your choice:1
Enter element to be insert:2
Enter your choice:1
Enter element to be insert:4
Enter your choice:1
Enter element to be insert:6
Enter your choice:3
        2       4       6
rear is at 6

front is at 2

Enter your choice:2
Deleted element is:2
Enter your choice:2
Deleted element is:4
Enter your choice:3
        6
rear is at 6

front is at 6

Enter your choice:1
Enter element to be insert:5
Enter your choice:3
        5       6
rear is at 5

front is at 6
```

QUESTION 2:

2. Menu-driven C program to perform insertion, deletion, search and display operations in an ordered list (ordered / sorted array).

PSEUDO CODE:

```
define max, list[MAX],i,len
int search(int),void insert(int),void del(int),void display()
read input n
do
if(n=1)
  read input e, insert(e)
```

```
else if(n=2)
  read input e, del(e)
else if(n=3)
  read input e, search(e)
else if(n=4) display()

int search(int e)
for(i=0;i<len;i++)
if(list[i]==e)
  print "The element is present at position:" i+1
if(i=len)  print "The search is unsuccessful"

void insert(int e)
if(len=MAX)
  print "LIST FULL!"
else p=len;
for(i=0;i<len;i++)
     if(list[i]>e)
          p=i
for(i=len-1;i>=p;i--)
     list[i+1]=list[i];
     list[p]=e;
     len++

void del(int e)
for(i=0;i<len;i++)
     if (list[i]==e)
     pos=i;
for(j=pos;j<len;j++)
     list[j]=list[j+1];
     len--

void display()
if(len)
     for(i=0;i<len;i++)
          print list[i]
else print "List empty"
```

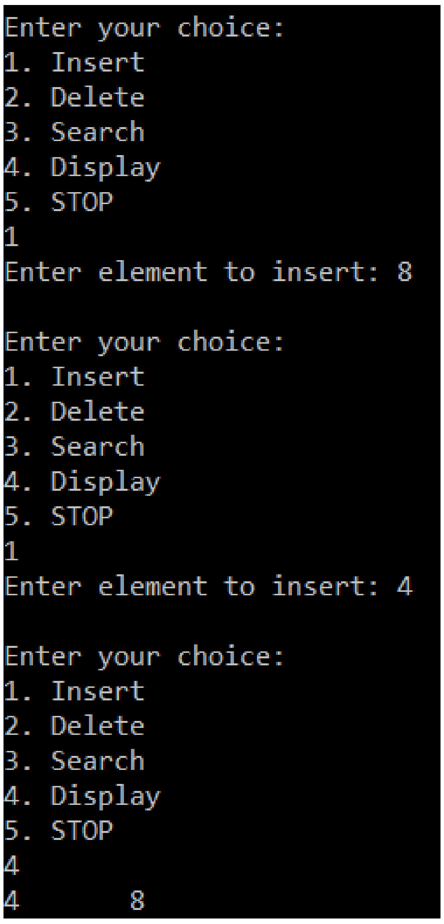
PROGRAM CODE:

```
#include<stdio.h>
#define MAX 5
int list[MAX];
int len=0,i,j,k,pos;
int search(int);
void insert(int);
void del(int);
void display();
main(){
int n;
do{
printf("\nEnter your choice: \n1. Insert\n2. Delete\n3.
Search\n4. Display\n5. STOP \n");
scanf("%d",&n);
if(n==1){
int e;
printf("Enter element to insert: ");
scanf("%d",&e);
insert(e);
}
else if(n==2){
int e;
printf("Enter element to be deleted: ");
scanf("%d",&e);
del(e);
}
else if(n==3){
int e;
printf("\nEnter the element to be searched: ");
scanf("%d",&e);
search(e);
}
else if(n==4){
display();
}
else if(n==5)
 break;
}while(1);
```

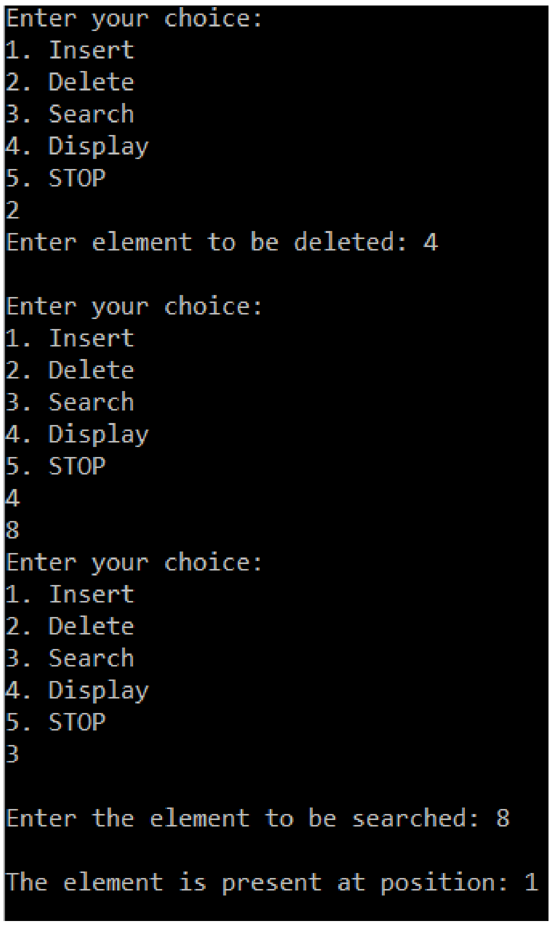
```
}
int search(int e){
for(i=0;i<len;i++){
if(list[i]==e)
 {
 printf("\nThe element is present at position: %d \n\n",i+1);
 break;
 }
 }
 if(i==len)
 {
 printf("\nThe search is unsuccessful");
 }
}
void insert(int e){
if(len==MAX)
    printf("LIST FULL!\n");
else{
    int p=len;
    for(i=0;i<len;i++){
    if(list[i]>e){
    p=i;
    break;
     }
}
for(i=len-1;i>=p;i--)
   list[i+1]=list[i];
   list[p]=e;
   len++;
   }
}
void del(int e){
for(i=0;i<len;i++){
if (list[i]==e)
pos=i;
}
for(j=pos;j<len;j++)
list[j]=list[j+1];
len--;
```

```
}
void display(){
if(len){
for(i=0;i<len;i++){
   printf("%d\t",list[i]);
   }
}
else
   printf("List empty\n");
}
```

PROGRAM OUTPUT:



```
Enter your choice:
1. Insert
2. Delete
3. Search
4. Display
5. STOP
1
Enter element to insert: 8

Enter your choice:
1. Insert
2. Delete
3. Search
4. Display
5. STOP
1
Enter element to insert: 4

Enter your choice:
1. Insert
2. Delete
3. Search
4. Display
5. STOP
4
4       8
```



```
Enter your choice:
1. Insert
2. Delete
3. Search
4. Display
5. STOP
2
Enter element to be deleted: 4

Enter your choice:
1. Insert
2. Delete
3. Search
4. Display
5. STOP
4
8
Enter your choice:
1. Insert
2. Delete
3. Search
4. Display
5. STOP
3

Enter the element to be searched: 8

The element is present at position: 1
```

QUESTION 3:

3. Menu-driven C program to perform insertion, deletion, search and display operations in an unordered list (unordered / unsorted array).

PSEUDO CODE:

```
define max, list[MAX],i,len=0,e,p
int search(int),void insert(int,int),void del(int),void
display()
do
read input n
if(n=1)
  read inputs e,p
  insert(e,p-1)
else if(n=2)
  read input e
  del(e-1)
else if(n=3)
  read input e
  search(e)
else if(n=4) display()

void insert(int e,int p)
if (len==MAX or (p<0 or p>len))
                 print "LIST FULL! OR Invalid position "
else
  for(i=len-1;i>=p;i--)
                 list[i+1]=list[i];
                 list[p]=e;
                 len++

void del(int p)
if((p<0 or p>=len) or len==0)
  print "Invalid position OR list empty "
else
                 for(i=p+1;i<len;i++)
                 list[i-1]=list[i];
                 len--

void display()
if(len)
                 for(i=0;i<len;i++)
                 print list[i]
else
```

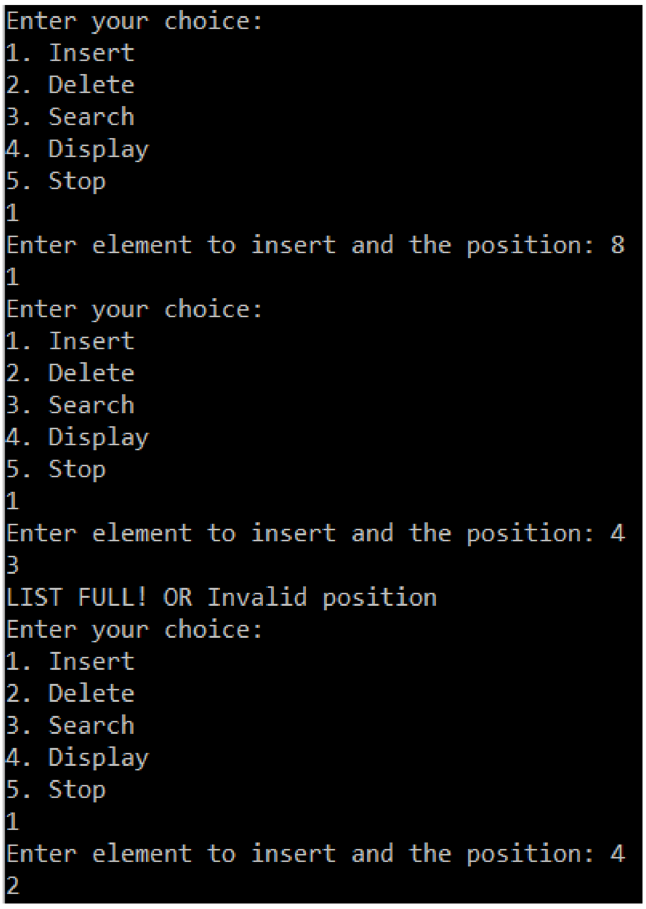
```
                 print "List empty"

int search(int e)
for(i=0;i<len;i++)
                 if(list[i]==e)
                   print "The element is present at
position:",i+1
                 if(i==len)
                   print "The search is unsuccessful"
```

PROGRAM CODE:

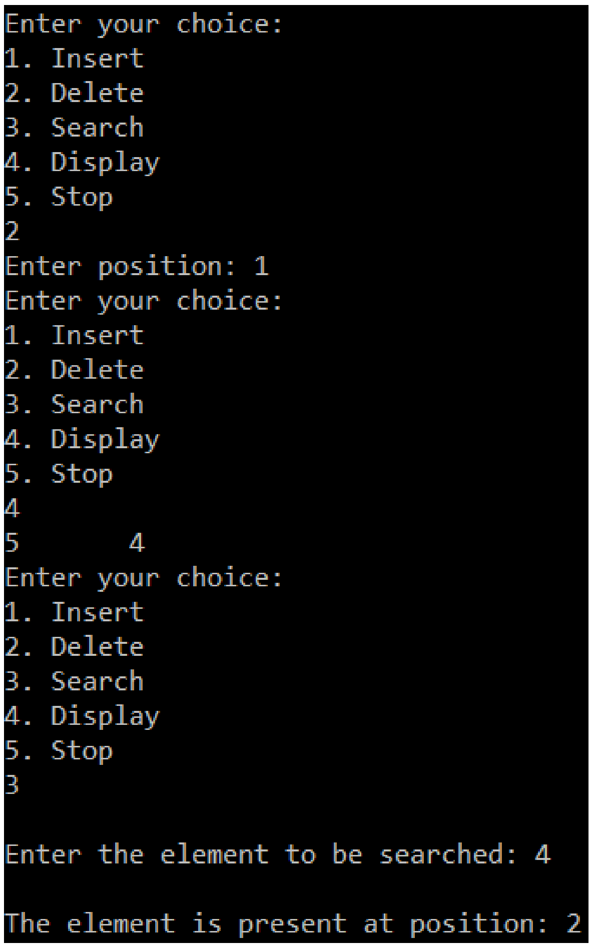
```
#include<stdio.h>
#define MAX 5
int list[MAX];
int len=0,i;
int search(int);
void insert(int,int);
void del(int);
void display();
main(){
int n;
do{
printf("Enter your choice: \n1. Insert\n2. Delete\n3. Search
\n4. Display \n5. Stop\n");
scanf("%d",&n);
if(n==1){
int e,p;
printf("Enter element to insert and the position: ");
scanf("%d %d",&e,&p);
insert(e,p-1);
}
else if(n==2){
int e;
printf("Enter position: ");
scanf("%d",&e);
del(e-1);
}
else if(n==3){
int e;
printf("\nEnter the element to be searched: ");
scanf("%d",&e);
search(e);
}
else if(n==4){
display();
}
else if(n==5)
 break;
}while(1);
}
void insert(int e,int p){
if(len==MAX || (p<0||p>len))
printf("LIST FULL! OR Invalid position\n");
else{
for(i=len-1;i>=p;i--)
    list[i+1]=list[i];
    list[p]=e;
    len++;
    }
}
void del(int p){
if((p<0||p>=len) || len==0)
   printf("Invalid position OR list empty\n");
else{
for(i=p+1;i<len;i++)
    list[i-1]=list[i];
    len--;
    }
}
void display(){
if(len){
for(i=0;i<len;i++)
   printf("%d ",list[i]);
   printf("\n");
}
else
  printf("List empty\n");
}
```

```
int search(int e){
for(i=0;i<len;i++){
  if(list[i]==e)
  {
   printf("\nThe element is present at position: %d \n",i);
   break;
  }
  }
   if(i==len)
   {
    printf("\nThe search is unsuccessful\n");
   }
}
```

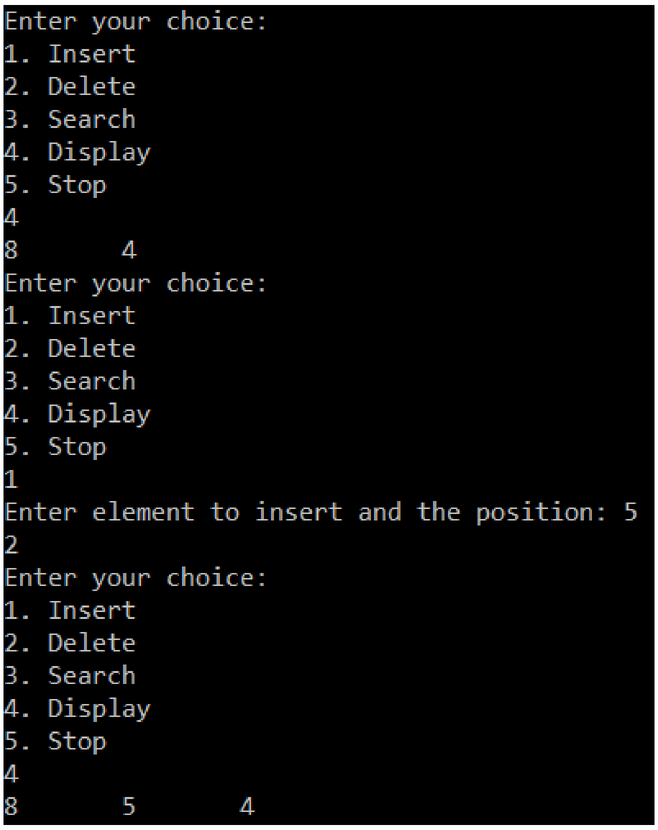
PROGRAM OUTPUT:



```
Enter your choice:
1. Insert
2. Delete
3. Search
4. Display
5. Stop
1
Enter element to insert and the position: 8
1
Enter your choice:
1. Insert
2. Delete
3. Search
4. Display
5. Stop
1
Enter element to insert and the position: 4
3
LIST FULL! OR Invalid position
Enter your choice:
1. Insert
2. Delete
3. Search
4. Display
5. Stop
1
Enter element to insert and the position: 4
2
```



```
Enter your choice:
1. Insert
2. Delete
3. Search
4. Display
5. Stop
2
Enter position: 1
Enter your choice:
1. Insert
2. Delete
3. Search
4. Display
5. Stop
4
5       4
Enter your choice:
1. Insert
2. Delete
3. Search
4. Display
5. Stop
3

Enter the element to be searched: 4

The element is present at position: 2
```



```
Enter your choice:
1. Insert
2. Delete
3. Search
4. Display
5. Stop
4
8       4
Enter your choice:
1. Insert
2. Delete
3. Search
4. Display
5. Stop
1
Enter element to insert and the position: 5
2
Enter your choice:
1. Insert
2. Delete
3. Search
4. Display
5. Stop
4
8       5       4
```

QUESTION 4:

4. Menu driven C program to implement singly linked list. Menu should have

   The following operations:

   a. Insertion

      i. Beginning insertion
      ii. End insertion
      iii. Position insertion

   b. Deletion

      i. Beginning deletion
      ii. End deletion
      iii. Position deletion

   c. Search

   d. Display

   e. Exit

PSEUDO CODE:

```
Struct node
{
        int data;
        struct node *next; //int a; int *p; //char c; char*p1;
}*list = 0;

int main()
int ch
void insert_beg()
void insert_end()
int insert_pos()
void display()
void delete_beg()
void delete_end()
int delete_pos()
int search()
read input ch
switch(ch)
case 1:
read input ch
  switch(ch)    //insert menu
                 case 1: insert_beg()
                 case 2: insert_end()
                 case 3: insert_pos()
                 case 4: exit(0)
case 2:
scan ch
  switch(ch) //Delete Menu
                 case 1: delete_beg()
                 case 2: delete_end()
                 case 3: delete_pos()
                 case 4: exit(0)
case 3: display()
case 4:
int index
index=search()
                 if(index!=-1)
                 print "element found at position":index+1
                 else print "element not found"
case 5: exit(0)

void insert_beg()
Read input data val
nnode = malloc(sizeof(struct node)
nnode -> data = val;
nnode -> next = 0;
                 if (list == 0)
                   list = nnode;
                 else
                   nnode -> next = list;
                   list = nnode;


void insert_end()
Read input data val
nnode = malloc(sizeof(struct node)
nnode -> data = val;
nnode -> next = 0;
                 if (list == 0)
                   list = nnode;
                 else
                    t = list;
                    while (t -> next != 0)
                        t = t->next;
                    t-> next = nnode;

int insert_pos()
Read input data val, pos
p = list
ctr = 1
While(ctr < pos-1) && (p -> next != 0)
{
     p = p -> next
     ctr++
}
                 nnode = malloc(sizeof(struct node)
                 nnode -> data = val
                 nnode -> next = p -> next
                 p -> next = nnode;

void display()
  if (list==0)
                 print “Empty List”
  else {
                 while (list != 0)
                   Print (list -> data)
                   list = list ->next
       }

void delete_beg()
if (list = 0)
                 print “empty list”
else {
                 Read deletion data d
                 if (list -> data = d) //Beginning deletion
                   print deletion data is list -> data
                   list = list -> next
                 else // Traverse and delete
                   t = list
                   while (t ->next != 0) and (t -> data != d)
                        s = t
                        t = t->next
                   if (t -> data = d) s -> next = t -> next
                   else print Deletion item not found
      }


void delete_end()
if(start==NULL)
         print "The list is empty!!"
    else{
         q=start;
         while(q->next->next!=NULL)
                 q=q->next;
                 t=q->next;
                 q->next=NULL;
                 print "Deleted element is",t->data
                 free(t);
         }

int delete_pos()
int pos,i
if(start=NULL)
     print "List is empty!!"
read input pos
q=start
for(i=1;i<pos-1;i++)
{
                 if(q->next=NULL)
                 {
                   print "There are less elements!!"
                 }
                 q=q->next;
}
t=q->next
q->next=t->next;
print "Deleted element is",t->data
free(t);

int search()
int key,index
read input key
struct node *curNode;
index = 0;
curNode = start;
// Iterate till last element until key is not found
while (curNode != NULL && curNode->data != key)
{
                 index++
                 curNode = curNode->next;
   }
                 return (curNode != NULL) ? index : -1;
```

PROGRAM CODE:

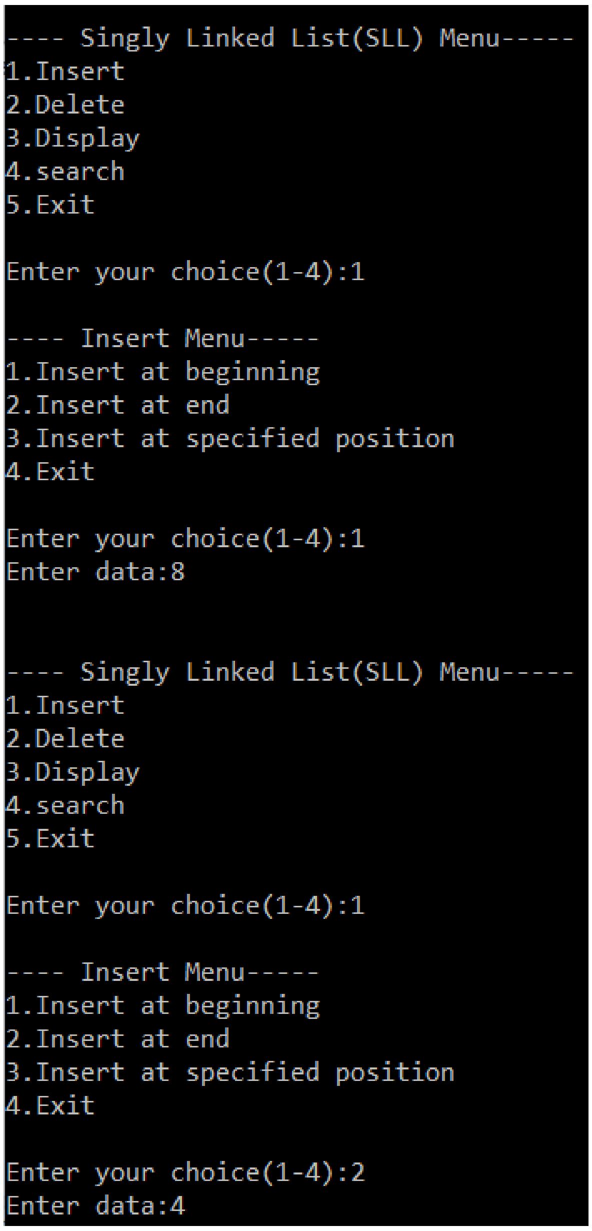
```
#include<stdio.h>
#include<conio.h>
#include<process.h>
struct node
{
int data;
struct node *next;
}*start=NULL,*q,*t;
int main()
{
int ch;
void insert_beg();
void insert_end();
int insert_pos();
void display();
void delete_beg();
void delete_end();
int delete_pos();
int search();
while(1)
{
printf("\n\n---- Singly Linked List(SLL) Menu-----");
printf("\n1.Insert\n2.Delete\n3.Display\n4.search\n5.Exit\n\n")
;
printf("Enter your choice(1-4):");
scanf("%d",&ch);
switch(ch)
{
case 1:
printf("\n---- Insert Menu-----");
printf("\n1.Insert at beginning\n2.Insert at end\n3.Insert at
specified position\n4.Exit");
printf("\n\nEnter your choice(1-4):");
scanf("%d",&ch);
switch(ch)
{
case 1: insert_beg();
break;
case 2: insert_end();
break;
case 3: insert_pos();
break;
case 4: exit(0);
default: printf("Wrong Choice!!");
}
break;
case 2: printf("\n---- Delete Menu-----");
printf("\n1.Delete from beginning\n2.Delete from end\n3.Delete
from specified position\n4.Exit");
printf("\n\nEnter your choice(1-4):");
scanf("%d",&ch);
switch(ch)
{
case 1: delete_beg();
break;
case 2: delete_end();
break;
case 3: delete_pos();
break;
case 4: exit(0);
default: printf("Wrong Choice!!");
}
break;
case 3: display();
break;
case 4:{int index;
index=search();
if(index!=-1)
printf("element found at position : %d",index+1);
else printf("element not found");}break;
case 5: exit(0);
default: printf("Wrong Choice!!");
    }
}
return 0;
}
void insert_beg()
```

```
{
int num;
t=(struct node*)malloc(sizeof(struct node));
printf("Enter data:");
scanf("%d",&num);
t->data=num;
if(start==NULL) //If list is empty
{
t->next=NULL;
start=t;
}
else
{
t->next=start;
start=t;
    }
}
void insert_end()
{
int num;
t=(struct node*)malloc(sizeof(struct node));
printf("Enter data:");
scanf("%d",&num);
t->data=num;
t->next=NULL;
if(start==NULL) //If list is empty
{
start=t;
}
else
{
q=start;
while(q->next!=NULL)
q=q->next;
q->next=t;
}
}
int insert_pos()
{
```

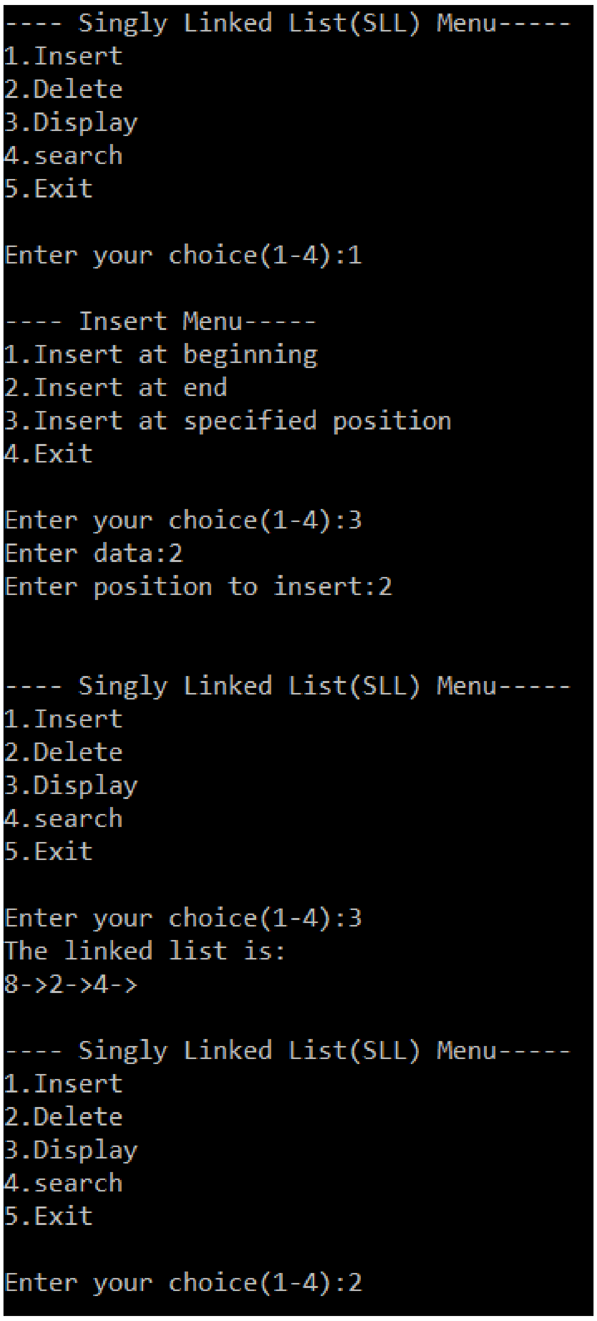
```
int pos,i,num;
if(start==NULL)
{
printf("List is empty!!");
return 0;
}
t=(struct node*)malloc(sizeof(struct node));
printf("Enter data:");
scanf("%d",&num);
printf("Enter position to insert:");
scanf("%d",&pos);
t->data=num;
q=start;
for(i=1;i<pos-1;i++)
{
if(q->next==NULL)
{
printf("There are less elements!!");
return 0;
}
q=q->next;
}
t->next=q->next;
q->next=t;
return 0;
}
void display()
{
if(start==NULL)
{
printf("List is empty!!");
}
else
{
q=start;
printf("The linked list is:\n");
while(q!=NULL)
{
printf("%d->",q->data);
q=q->next;
      }
   }
}
void delete_beg()
{
if(start==NULL)
{
printf("The list is empty!!");
}
else
{
q=start;
start=start->next;
printf("Deleted element is %d",q->data);
free(q);
     }
}
void delete_end()
{
if(start==NULL)
{
printf("The list is empty!!");
}
else
{
q=start;
while(q->next->next!=NULL)
q=q->next;
t=q->next;
q->next=NULL;
printf("Deleted element is %d",t->data);
free(t);
    }
}
int delete_pos()
{
int pos,i;
if(start==NULL)
```

```
{
printf("List is empty!!");
return 0;
}
printf("Enter position to delete:");
scanf("%d",&pos);
q=start;
for(i=1;i<pos-1;i++)
{
if(q->next==NULL)
{
printf("There are less elements!!");
return 0;
}
q=q->next;
}
t=q->next;
q->next=t->next;
printf("Deleted element is %d",t->data);
free(t);
return 0;
}
int search()
{
int key;
printf("enter element to be searced : ");
scanf("%d",&key);
int index;
struct node *curNode;
index = 0;
curNode = start;
while (curNode != NULL && curNode->data != key)
{
index++;
curNode = curNode->next;
}
return (curNode != NULL) ? index : -1;}
```

PROGRAM OUTPUT:

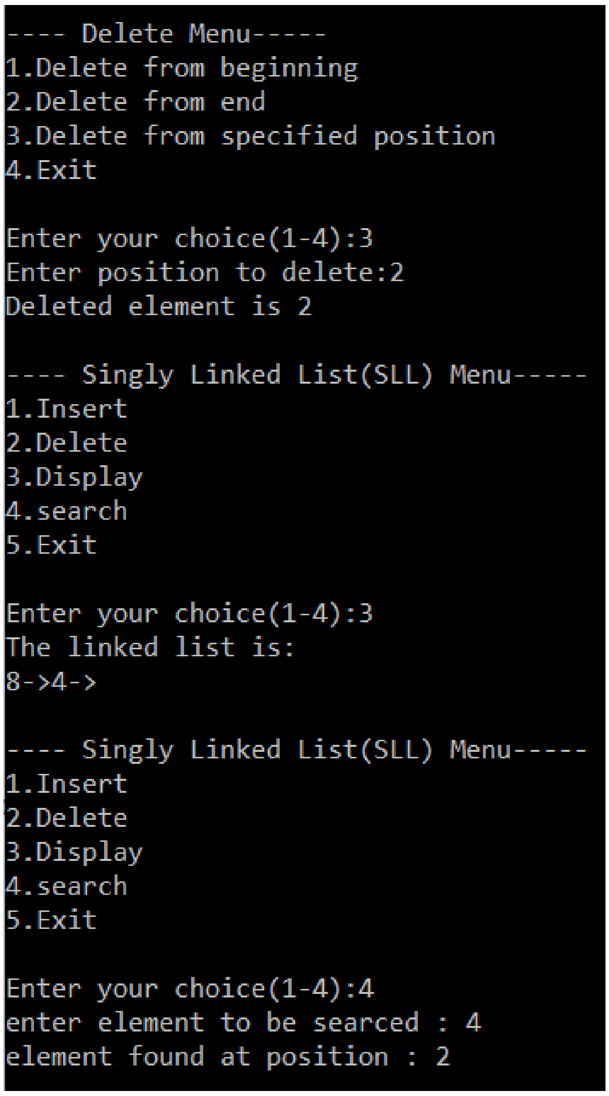


```
---- Singly Linked List(SLL) Menu-----
1.Insert
2.Delete
3.Display
4.search
5.Exit

Enter your choice(1-4):1

---- Insert Menu-----
1.Insert at beginning
2.Insert at end
3.Insert at specified position
4.Exit

Enter your choice(1-4):1
Enter data:8


---- Singly Linked List(SLL) Menu-----
1.Insert
2.Delete
3.Display
4.search
5.Exit

Enter your choice(1-4):1

---- Insert Menu-----
1.Insert at beginning
2.Insert at end
3.Insert at specified position
4.Exit

Enter your choice(1-4):2
Enter data:4
```



```
---- Singly Linked List(SLL) Menu-----
1.Insert
2.Delete
3.Display
4.search
5.Exit

Enter your choice(1-4):1

---- Insert Menu-----
1.Insert at beginning
2.Insert at end
3.Insert at specified position
4.Exit

Enter your choice(1-4):3
Enter data:2
Enter position to insert:2


---- Singly Linked List(SLL) Menu-----
1.Insert
2.Delete
3.Display
4.search
5.Exit

Enter your choice(1-4):3
The linked list is:
8->2->4->

---- Singly Linked List(SLL) Menu-----
1.Insert
2.Delete
3.Display
4.search
5.Exit

Enter your choice(1-4):2
```

19BCE0829
KANDRA KSHEERAJ



```
---- Delete Menu-----
1.Delete from beginning
2.Delete from end
3.Delete from specified position
4.Exit

Enter your choice(1-4):3
Enter position to delete:2
Deleted element is 2

---- Singly Linked List(SLL) Menu-----
1.Insert
2.Delete
3.Display
4.search
5.Exit

Enter your choice(1-4):3
The linked list is:
8->4->

---- Singly Linked List(SLL) Menu-----
1.Insert
2.Delete
3.Display
4.search
5.Exit

Enter your choice(1-4):4
enter element to be searced : 4
element found at position : 2
```

- -