

*Untitled - Notepad

File Edit Format View Help

```
#include<stdio.h>
#include<malloc.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;
}
int freenode(NODE x)
{
    free(x);
    return 0;
}
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;
}
```

*Untitled - Notepad

File Edit Format View Help

```
}
```

```
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;
```

```
}
```

```
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);
```

*Untitled - Notepad

File Edit Format View Help

```
printf("%d\n",temp->info);
}
}
NODE delete_pos(int pos, NODE first)
{
NODE cur;
NODE prev;
int count;
if(first==NULL || pos<=0)
{
printf("invalid position \n");
return NULL;
}
if (pos==1)
{
cur=first;
first=first->link;
freenode(cur);
return first;
}
prev=NULL;
cur=first;
count=1;
while(cur!=NULL)
{
if(count==pos) break;
prev=cur;
cur=cur->link;
count++;
}
if(count!=pos)
{
printf("invalid position \n");
return first;
}
if(count!=pos)
{
printf("invalid position specified \n");
return first;
}
prev->link=cur->link;
freenode(cur);
return first;
}
int main()
{
int item,choice,pos;
NODE first=NULL;
system("cls");
for(...)
```

*Untitled - Notepad

File Edit Format View Help

```
}

if(count!=pos)
{
printf("invalid position specified \n");
return first;
}
prev->link=cur->link;
freenode(cur);
return first;
}
int main()
{
int item,choice,pos;
NODE first=NULL;
system("cls");
for(;;)
{
printf("\n 1:Insert_front\n 2:Delete_front\n 3:Insert_rear\n 4:Delete_rear\n 5.Delete at specified position \n 6:Display_list\n 7: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: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 position of the item to be deleted: \n");
      scanf("%d",&pos);
      first=delete_pos(pos,first);
      break;

case 6:display(first);
break;
default:exit(0);
break;
}
}
getch();
return 0;
}
```

1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5.Delete at specified position
6:Display_list
7:Exit

enter the choice

1

enter the item at front-end

9

1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5.Delete at specified position
6:Display_list
7:Exit

enter the choice

1

enter the item at front-end

6

I

1:Insert_front
2:Delete_front
3:Insert_rear
4:Delete_rear
5.Delete at specified position
6:Display_list
7:Exit

enter the choice



```
1:Insert_front  
2:Delete_front  
3:Insert_rear  
4:Delete_rear  
5.Delete at specified position  
6:Display_list  
7:Exit  
enter the choice  
2  
item deleted at front-end is=6
```

```
1:Insert_front  
2:Delete_front  
3:Insert_rear  
4:Delete_rear  
5.Delete at specified position  
6:Display_list  
7:Exit  
enter the choice  
4  
item deleted is 9
```

```
1:Insert_front  
2:Delete_front  
3:Insert_rear  
4:Delete_rear  
5.Delete at specified position  
6:Display_list  
7:Exit  
enter the choice
```

LAB PROGRAM - 5

```
#include <stdio.h>
#include <malloc.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);
    }
    else
        x;
    return x;
}

int pnode (NODE x)
{
    if (x)
        printf ("%d", x->info);
    return 0;
}

NODE insert (NODE front, int item)
{
    NODE temp;
    temp = getnode();
    temp->info = item;
    temp->link = front;
    if (front == NULL)
        front = temp;
    return front;
}
```

return temp;

temp \rightarrow link = first;
first = temp;
return first;

}

NOTE delete - first (NODE first)

{

NODE temp;

if (first == NULL)

{

print ("List is empty\n");
return first;

}

temp = first;

temp = temp \rightarrow link;

print ("Item deleted at pos and is = %d\n",
first \rightarrow info);

free (first);

return temp;

}

NOTE insert - node (NODE first, int it)

{

NODE temp, cur;

temp = getNode();

temp \rightarrow info = it;

temp \rightarrow link = cur;

if (first == NULL)

return temp;

(cur = first;

while ($\text{Cur} \geq (\text{inR} = \text{NoCCL})$)

$\text{Cur} = \text{Cur} - 1$;

$\text{Cur} \rightarrow (\text{ind} = \text{start};$

return first;

}

2. NODE delete - ~~Zero~~ (More ~~first~~)

2

NODE Cur, prev;

if ($\text{Cur} = \text{NoCCL}$)

1

print ("list is empty. Cannot delete in");
return first;

3

if ($\text{first} \rightarrow (\text{inR} = \text{NoCCL})$)

{

printf ("item deleted is '%c'\n", first->info);
free (first);

return NoCCL;

}

prev = NoCCL;

cur = first;

while ($\text{Cur} \geq \text{link}! = \text{NoCCL}$)

{

prev = cur;

cur = cur - link;

3

printf ("item deleted at index %d; cur=%c\n",

free (cur);

(prev->link = NoCCL);

return first;

4

void display (NODE first) {

8

NOTE Tony;

if Cpos = = NULL L)

print of C" list empty cannot display if one (");

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

{

print of C" /d in", temp -> info);

}

3

NOTE Cor;

NOTE prev;

int Count;

if (Cpos == NULL & pos <= 0)

print of C" invalid position (");

return NULL;

4

if Cpos == 0

5

Cur = first;

first = first->link;

freemem(Cur);

return first;

3

if cur == NULL;

cur = first

Count = 1;

while (Cur != NULL)

6

if (Count == pos) break;

prevCur;

Cur = Cur->link

Count ++;

3
your \rightarrow lim b = cur > limb;
SCREEN de (cur);
return first;

3
int main()

{
int item, choice, pos;
NODE first = NULL;
Splay C "CLSL";
for (c)
3

{ if (C ==

Y
choice C "In 1:Insert :front\n": Delete front \n
3. Insert, Y or N 4:Delete, X or Z 5:Delete at Specified
position In 6: Display list In 7: Exit In"\n;
Update of C "Enter the choice In"\n;
Scan C "/.d", & choice"\n";
Switch (choice)

2

Case 1: printf C "Enter the item at front - and \n";
Scan C "/.d", & item;
first = insertFront(first, item);
break;

Case 2: first = deleteFront(first);
break;

Case 3: printf C "Enter the item address - and \n";
Scan C "/.d", & item;
first = insertAfter(first, item);
break;

Case 5: printf C "Enter the position of the item to be deleted : \n";

Scanf C11/d, & pos);

first = delete - pos (pos, first);
break;

case 6: display (first);

break;

default: exit (0);

break;

}

3

getch();

return 0;

.