

*Untitled - Notepad

File Edit Format View Help

```
#include<stdio.h>
#include<conio.h>
#include<malloc.h>
#include<process.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_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_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 and is-%d\n",temp->info);
```

*Untitled - Notepad

File Edit Format View Help

```
printf("item deleted at front-end is=%d\n",first->info);
free(first);
return temp;
}
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);
}
}
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_s(NODE first)
{
NODE temp;
if(first==NULL)
{
printf("stack 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;
}
void display_s(NODE first)
{
NODE temp;
if(first==NULL)
printf("stack 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

```
free(first);
return temp;
}
void display_s(NODE first)
{
NODE temp;
if(first==NULL)
printf("stack 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;
system("cls");
for(;;)
{
printf("\n Queue operations :\n 1:Insert_rear\n 2:Delete_front\n 3:Display_list(Queue)\n \n Stack operations \n
4:Insert_front\n 5: Delete_front \n 6:Display_list(Stack)\n 7:Exit \n \n");
printf("enter the choice \n");
scanf("%d",&choice);
switch(choice)
{
case 1:printf("enter the item at rear-end\n");
scanf("%d",&item);
first=insert_rear(first,item);
break;
case 2:first=delete_front(first);
break;
case 3:display(first);
break;
case 4:printf("enter the item at front-end\n");
scanf("%d",&item);
first=insert_front(first,item);
break;
case 5:first=delete_front_s(first);
break;
case 6:display_s(first);
break;
default:exit(0);
break;
}
}
getch();
return 0;
}
```

```
Queue operations :  
1:Insert_rear  
2:Delete_front  
3:Display_list(Queue)
```

```
Stack operations  
4:Insert_front  
5: Delete front  
6:Display_list(Stack)  
7:Exit
```

enter the choice

1

enter the item at rear-end

23

```
Queue operations :  
1:Insert_rear  
2:Delete_front  
3:Display_list(Queue)
```

```
Stack operations  
4:Insert_front  
5: Delete front  
6:Display_list(Stack)  
7:Exit
```

enter the choice

1

enter the item at rear-end

67

```
Queue operations :
```

```
Queue operations :  
1:Insert_rear  
2:Delete_front  
3:Display_list(Queue)
```

```
Stack operations  
4:Insert_front  
5: Delete_front  
6:Display_list(Stack)  
7:Exit
```

enter the choice

2
item deleted at front-end is=23

```
Queue operations :  
1:Insert_rear  
2:Delete_front  
3:Display_list(Queue)
```

```
Stack operations  
4:Insert_front  
5: Delete_front  
6:Display_list(Stack)  
7:Exit
```

enter the choice

3
67

```
Queue operations :  
1:Insert_rear  
2:Delete_front
```

```
Stack operations
4:Insert_front
5: Delete_front
6:Display_list(Stack)
7:Exit
```

enter the choice

4

enter the item at front-end

58

```
Queue operations :
```

```
1:Insert_rear
2:Delete_front
3:Display_list(Queue)
```

```
Stack operations
```

```
4:Insert_front
5: Delete_front
6:Display_list(Stack)
7:Exit
```

enter the choice

4

enter the item at front-end

89

```
Queue operations :
```

```
1:Insert_rear
2:Delete_front
3:Display_list(Queue)
```

```
Stack operations
```

Stack operations
4:Insert_front
5: Delete_front
6:Display_list(Stack)
7:Exit

enter the choice

5
item deleted at front-end is=89

Queue operations :
1:Insert_rear
2:Delete_front
3:Display_list(Queue)

Stack operations
4:Insert_front
5: Delete_front
6:Display_list(Stack)
7:Exit

enter the choice

6
58
67

Queue operations :
1:Insert_rear
2:Delete_front
3:Display_list(Queue)

Stack operations
4:Insert_front

LAB PROGRAM - 8

```
#include <stdio.h>
#include <conio.h>
#include <malloc.h>
#include <process.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)
    {
        primey ("Memory full\n");
        exit(0);
    }
    return x;
}

Void freeNode (NODE A)
{
    free (A);
}

NODE insert_order (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_front (NODE first)
{
    NODE temp;
    temp = first->link;
    free (first);
    return temp;
}
```

```
{ NODE temp;  
if (first == NULL)  
{ cout << "list is empty cannot delete \n";  
return first; }  
temp = first;  
temp = temp->link;  
cout << "item deleted at position and is = " << first->info  
<< endl;  
first = temp; }  
void display (NODE first)  
{ NODE temp;  
if (first == NULL)  
cout << "list is empty cannot display items \n";  
else { temp = first; temp->NULL; temp = temp->link; }  
{ cout << first->info << endl; temp = temp->link; } }  
NODE insert (NODE first, int item)  
{ NODE temp;  
temp = getnode();  
temp->info = item;  
temp->link = NULL;  
if (first == NULL)  
return temp;  
temp->link = first;  
first = temp; }  
NODE delete (NODE first)  
{ NODE temp;  
if (first == NULL)  
cout << "list is empty cannot delete \n";  
else { temp = first; }  
temp = temp->link;
```

printf("item deleted at front and is = %d\n", front->info);
front (choose);
return item; } }

void display => C NODE first
{ NODE temp;

if (first == NULL)

printf ("Stack empty. Cannot display items\n").
yes (temp = first; temp != NULL; temp = temp->link)
{ printf ("%d ", temp->info); } }
int main()

{ int item, choice, pos;

NODE first = NULL;

System ("cls");

choice(); }

choice if ("1: Insert item in 2: Delete
front in 3: Display list (queue) in 4: Stack operations
5: Insert front in 6: Delete front in 7: Display list (stack) in
8: Exit in 0");

choice if ("enter the choice in");

Scangf ("%d", &choice);

switch (choice)

{ Case 1: choice if ("enter the item at queue - and in");

Scangf ("%d", &item);

front = insert_rear (front, item);

break;

Case 2: front = delete_front (front);

break;

Case 3: display = (front);

break;

Case 4: printf ("enter the item at front - and in");

Scangf ("%d", &item);

front = insert_front (front, item);

break;

case 5: first = delete; front = first;

break;

case 6: display &(first);

break;

default: exit(0);

break;

if (t == 0)

3

getch();

otherwise;

3