S.R.POOJA
1BM19CS135

```
#include <stdio.h>
#include <stdlib.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; }
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;
return first; }
NODE delete_front (NODE first) {
NODE temp;
if (first== NULL) {
printf ("list is empty cannot delete\n");
return first; }
temp= first;
```

S.R.POOJA
1BM19CS135

```
temp= temp->link;
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 delete 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;
return 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); }}
```

S. R. POOJA
1BM19CS135

```
int main(){
int item, choice, pos;
NODE first=NULL;
system("cls");
for(;;)
{ printf("\n Queue operations :\n 1. Insert_rear \n 2. Delete_f
\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;
}
```

```c
#include<stdio.h>
#include<stdlib.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;
}
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");
```

```
45  return first;
46  }
47  temp=first;
48  temp=temp->link;
49  printf("item deleted at front-end is=%d\n",first->info);
50  free(first);
51  return temp;
52  }
53  void display(NODE first)
54  {
55  NODE temp;
56  if(first==NULL)
57  printf("list empty cannot display items\n");
58  for(temp=first;temp!=NULL;temp=temp->link)
59  {
60  printf("%d \n",temp->info);
61  }
62  }
63  NODE insert_front(NODE first,int item)
64  {
65  NODE temp;
66  temp=getnode();
67  temp->info=item;
68  temp->link=NULL;
69  if(first==NULL)
70  return temp;
71  temp->link=first;
72  first=temp;
73  return first;
74  }
75  NODE delete_front_s(NODE first)
76  {
77  NODE temp;
78  if(first==NULL)
79  {
80  printf("stack is empty cannot delete\n");
81  return first;
82  }
83  temp=first;
84  temp=temp->link;
85  printf("item deleted at front-end is=%d\n",first->info);
86  free(first);
87  return temp;
88  }
```

```
92   if(first==NULL)
93   printf("stack empty cannot display items\n");
94   for(temp=first;temp!=NULL;temp=temp->link)
95   {
96   printf("%d\n",temp->info);
97   }
98   }
99   int main()
100  {
101  int item,choice,pos;
102  NODE first=NULL;
103  system("cls");
104  for(;;)
105  {
106  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:Dislay_list(Stack)\n 7:Exit \n \n");
107  printf("enter the choice \n");
108  scanf("%d",&choice);
109  switch(choice)
110  {
111  case 1:printf("enter the item at rear-end\n");
112  scanf("%d",&item);
113  first=insert_rear(first,item);
114  break;
115  case 2:first=delete_front(first);
116  break;
117  case 3:display(first);
118  break;
119  case 4:printf("enter the item at front-end\n");
120  scanf("%d",&item);
```

```
> clang-7 -pthread -lm -o main main.c
main.c:131:1: warning: implicit declaration of function 'getch' is invalid in
      C99 [-Wimplicit-function-declaration]
getch();
^
1 warning generated.
> ./main
sh: 1: cls: not found

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

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

enter the choice
1
enter the item at rear-end
12
```

```
enter the choice
1
enter the item at rear-end
12

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

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

enter the choice
3
12

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

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

```
enter the item at front-end
14

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

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

enter the choice
4
enter the item at front-end
17

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

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

enter the choice
6
17
14

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

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

enter the choice
```