

→ DS Lab Program 2 -

① #include <stdio.h>

#include <stdlib.h>

struct Node {

int info;

struct Node \* link;

};

typedef struct Node node;

node getnode () {

node n;

n = (node) malloc (sizeof (node));

if (n == NULL) {

printf ("Memory full\n");

exit (0);

}

return n;

}

void freenode (node n) {

free (n);

}

```
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 ("Queue empty cannot delete\n");
```

```
        return first;
```

```
    }
```

```
    temp = first;
```

```
    temp = temp -> link;
```

```
    printf ("item deleted at front = %d\n",  
            first -> info);
```

```
    free (first);
```

```
    return temp;
```

```
}
```

```
void display (node first) {
```

```
    node temp;
```

```
    if (first == NULL)
```

```
        printf ("Queue empty\n");
```

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

```
        printf ("%d\n", temp -> info);
```

```
}
```



```

    printf ("Enter 1 for que  

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;

```

4

```

int main() &  

    int item, choice, pos, c;  

    node first = NULL;  

for (c;;) printf ("Enter 1 for stack, 2 for  

    queue\n");  

    scanf ("%d", &c);  

    switch (c) &  

    case 1:  

        for (c;;) &  

            printf ("1. Insert front 2. Delete front 3. Display 4. exit\n");  

            printf ("Enter choice\n");  

            scanf ("%d", &choice);  

            switch (choice) &  

            case 1: printf ("Enter item\n");  

                scanf ("%d", &item);  

                first = insert-front (first, item);

```

```

        break;
    case 2: first = delete_front(first);
        break;
    case 3: display(first);
        break;
    default: exit(0);
        break;
}

```

```

}

```

```

break;

```

```

case 2: for(;;) {
    printf("\n 1. Insert rear\n 2. Delete front\n
        3. Display\n 4. Exit\n");

```

```

    scanf("%d", &choicechoice);

```

```

    switch(choice) {

```

```

        case 1: printf("Enter item\n");

```

```

            scanf("%d", &item);

```

```

            first = insert_rear(first, item);

```

```

            break;

```

```

        case 2: first = delete_front(first);

```

```

            break;

```

```

        case 3: display(first);

```

```

            break;

```

```

        default: exit(0);

```

```

            break;

```

```

}

```

```

break;

```

```

default: exit(0);

```

```

break;

```

```

}

```

```

}

```



```

1  #include<stdio.h>
2  #include<stdlib.h>
3  struct node {
4      int info;
5      struct node*link;
6  };
7  typedef struct node*NODE;
8  NODE getnode(){
9      NODE x;
10     x=(NODE)malloc(sizeof(struct node));
11     if(x==NULL) {
12         printf("memfull\n");
13         exit(0);
14     }
15     return x;
16 }
17 void freenode(NODE x){
18     free(x);
19 }
20 NODE insert_front(NODE first,int item) {
21     NODE temp;
22     temp=getnode();
23     temp->info=item;
24     temp->link=NULL;
25     if(first==NULL)
26         return temp;
27     temp->link=first;
28     first=temp;
29     return first;
30 }
31 NODE delete_front(NODE first){
32     NODE temp;
33     if(first==NULL) {
34         printf("stack is empty cannot delete\n");
35         return first;
36     }
37     temp=first;
38     temp=temp->link;
39     printf("item deleted at front-end is=%d\n",first->info);
40     free(first);
41     return temp;
42 }
43 void display(NODE first) {
44     NODE temp;
45     if(first==NULL)

```

```

42     free(first);
43     return temp;
44 }
45 void display(NODE first) {
46     NODE temp;
47     if(first==NULL)
48         printf("Queue empty cannot display items\n");
49     for(temp=first;temp!=NULL;temp=temp->link) {
50         printf("%d\n",temp->info);
51     }
52 }
53 int main() {
54     int item,choice,pos;
55     NODE first=NULL;
56     for(;;) {
57         printf("\n1:Insert_rear\n2>Delete_front\n3:Display_list\n4:Exit\n");
58         printf("enter the choice\n");
59         scanf("%d",&choice);
60         switch(choice) {
61             case 1:printf("enter the item at rear-end\n");
62                     scanf("%d",&item);
63                     first=insert_rear(first,item);
64                     break;
65             case 2:first=delete_front(first);
66                     break;
67             case 3:display(first);
68                     break;
69             default:exit(0);
70                     break;
71         }
72     }
73 }
74
75

```

```

1  #include<stdio.h>
2  #include<stdlib.h>
3  struct node {
4      int info;
5      struct node*link;
6  };
7  typedef struct node *NODE;
8  NODE getnode() {
9      NODE x;
10     x=(NODE)malloc(sizeof(struct node));
11     if(x==NULL) {
12         printf("mem full\n");
13         exit(0);
14     }
15     return x;
16 }
17 void freenode(NODE x) {
18     free(x);
19 }
20 NODE insert_rear(NODE first,int item) {
21     NODE temp,cur;
22     temp=getnode();
23     temp->info=item;
24     temp->link=NULL;
25     if(first==NULL)
26         return temp;
27     cur=first;
28     while(cur->link!=NULL)
29         cur=cur->link;
30     cur->link=temp;
31     return first;
32 }
33 NODE delete_front(NODE first) {
34     NODE temp;
35     if(first==NULL) {
36         printf("Queue is empty cannot delete\n");
37         return first;
38     }
39     temp=first;
40     temp=temp->link;
41     printf("item deleted at front-end is=%d\n",first->info);
42     free(first);
43     return temp;
44 }
45 void display(NODE first) {

```



```

43 void display(NODE first) {
44     NODE temp;
45     if(first==NULL)
46         printf("stack empty cannot display items\n");
47     for(temp=first;temp!=NULL;temp=temp->link) {
48         printf("%d\n",temp->info);
49     }
50 }
51 int main() {
52     int item,choice,pos;
53     NODE first=NULL;
54     for(;;) {
55         printf("\n1:Insert_front\n2>Delete_front\n3:Display_list\n4:Exit\n");
56         printf("enter the choice\n");
57         scanf("%d",&choice);
58         switch(choice) {
59             case 1:printf("enter the item at front-end\n");
60                     scanf("%d",&item);
61                     first=insert_front(first,item);
62                     break;
63             case 2:first=delete_front(first);
64                     break;
65             case 3:display(first);
66                     break;
67             default:exit(0);
68                     break;
69         }
70     }
71 }
72

```



```
1:Insert_rear
2:Delete_front
3:Display_list
4:Exit
enter the choice
1
enter the item at rear-end
10
```

```
1:Insert_rear
2:Delete_front
3:Display_list
4:Exit
enter the choice
1
enter the item at rear-end
20
```

```
1:Insert_rear
2:Delete_front
3:Display_list
4:Exit
enter the choice
1
enter the item at rear-end
30
```

```
1:Insert_rear
2:Delete_front
3:Display_list
4:Exit
enter the choice
3
10
20
30
```

```
1:Insert_rear
```

2:Delete\_front

3:Display\_list

4:Exit

enter the choice

2

item deleted at front-end is×10

1:Insert\_rear

2:Delete\_front

3:Display\_list

4:Exit

enter the choice

2

item deleted at front-end is×20

1:Insert\_rear

2:Delete\_front

3:Display\_list

4:Exit

enter the choice

2

item deleted at front-end is×30

1:Insert\_rear

2:Delete\_front

3:Display\_list

4:Exit

enter the choice

2

Queue is empty cannot delete

1:Insert\_rear

2:Delete\_front

3:Display\_list

4:Exit

enter the choice



```
1:Insert_front
2:Delete_front
3:Display_list
4:Exit
enter the choice
1
enter the item at front-end
10

1:Insert_front
2:Delete_front
3:Display_list
4:Exit
enter the choice
1
enter the item at front-end
20

1:Insert_front
2:Delete_front
3:Display_list
4:Exit
enter the choice
1
enter the item at front-end
30

1:Insert_front
2:Delete_front
3:Display_list
4:Exit
enter the choice
3
30
20
10

1:Insert_front
2:Delete_front
3:Display_list
4:Exit
enter the choice
2
item deleted at front-end is=30

1:Insert_front
2:Delete_front
3:Display_list
```

3:Display\_list

4:Exit

enter the choice

2

item deleted at front-end is=20

1:Insert\_front

2>Delete\_front

3:Display\_list

4:Exit

enter the choice

2

item deleted at front-end is=10

1:Insert\_front

2>Delete\_front

3:Display\_list

4:Exit

enter the choice

2

stack is empty cannot delete

1:Insert\_front

2>Delete\_front

3:Display\_list

4:Exit

enter the choice

3

stack empty cannot display items

1:Insert\_front

2>Delete\_front

3:Display\_list

4:Exit

enter the choice