→ DS Lab Program 8 -

①
```
#include <stdio.h>
#include <stdlib.h>
struct Node {
  int info;
  struct Node * link;
};
typedef struct Node node;
node getnode () {
  node x;
  x = (node) malloc (sizeof (node));
  if (x == NULL) {
    printf ("Memory 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("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 -> li
        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;
}
int main() {
int item, choice, pos, c;
node first = NULL;
printf("Enter 1 for stack, 2 for
queue\n");
scanf("%d", &c);
switch (c) {
case 1:
for (;;) {
printf("\n1: Insert front \n2. Delete front \n
3. Display \n4. exit");
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 ("\n1. Insert rear \n2. Delete front \n
                3. Display \n4. exit \n");
scanf ("%d", &choice);
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;
  }
}
```

```c
#include<stdio.h>
#include<stdlib.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("memfull\n");
        exit(0);
        }
    return x;
    }
void freenode(NODE x){
    free(x);
    }
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("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(NODE first) {
    NODE temp;
    if(first==NULL)
```

```
    free(first);
    return temp;
    }
void display(NODE first) {
    NODE temp;
    if(first==NULL)
        printf("Queue 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;
    for(;;) {
        printf("\n1:Insert_rear\n2:Delete_front\n3:Display_list\n4:Exit\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;
            default:exit(0);
                    break;
        }
    }
}
```

```c
#include<stdio.h>
#include<stdlib.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("Queue 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(NODE first) {
```

```c
void display(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;
    for(;;) {
        printf("\n1:Insert_front\n2:Delete_front\n3:Display_list\n4: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:display(first);
                   break;
            default:exit(0);
                   break;
        }
    }
}
```

```
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
```