```c
#include <stdio.h>
#include <stdlib.h>

struct Node {
    int data;
    struct Node* next;
};

struct Node* createNode(int data) {
    struct Node* newNode = (struct Node*) malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->next = NULL;
    return newNode;
}

void push(struct Node** top, int data) {
    struct Node* newNode = createNode(data);
    newNode->next = *top;
    *top = newNode;
}

int pop(struct Node** top) {
    if (*top == NULL) {
        printf("Stack underflow");
        return -1;
    } else {
        struct Node* temp = *top;
        *top = temp->next;
        int popped = temp->data;
        free(temp);
        return popped;
    }
}

int peek(struct Node* top) {
    if (top == NULL) {
        printf("Stack is empty");
        return -1;
    } else {
        return top->data;
    }
}

void dispStack(struct Node* top) {
```

```
    if (top == NULL) {
        printf("Stack is empty");
    } else {
        printf("Stack elements: ");
        struct Node* temp = top;
        while (temp != NULL) {
            printf("%d ", temp->data);
            temp = temp->next;
        }
        printf("\n");
    }
}

void enqueue(struct Node** f, struct Node** r, int data) {
    struct Node* newNode = createNode(data);
    if (*r == NULL) {
        *f = *r = newNode;
        return;
    }
    (*r)->next = newNode;
    *r = newNode;
}

int dequeue(struct Node** f) {
    if (*f == NULL) {
        printf("Underflow");
        return -1;
    } else {
        struct Node* temp = *f;
        int dequeued = temp->data;
        *f = temp->next;
        free(temp);
        return dequeued;
    }
}

void dispQueue(struct Node* f) {
    if (f == NULL) {
        printf("Queue is Empty!!");
    } else {
        printf("Queue elements: ");
        struct Node* temp = f;
        while (temp != NULL) {
            printf("%d ", temp->data);
            temp = temp->next;
```

```
        }
        printf("\n");
    }
}

int main() {
    struct Node* stack1 = NULL;
    struct Node *q1 = NULL, *q2 = NULL;
    int c, d;
    printf("Enter your choice:\n1. Push to Stack\n2. Pop from Stack\n3. Peek
Stack\n4. Display Stack\n5. Enqueue\n6. Dequeue\n7. Display Queue\n8. Exit\n");

    while (1) {
        printf("Enter your choice: ");
        scanf("%d", &c);
        switch (c) {
            case 1:
                printf("Enter data to push: ");
                scanf("%d", &d);
                push(&stack1, d);
                break;
            case 2:
                d = pop(&stack1);
                printf("Popped element is: %d\n", d);
                break;
            case 3:
                d = peek(stack1);
                printf("Top element: %d\n", d);
                break;
            case 4:
                dispStack(stack1);
                break;
            case 5:
                printf("Enter data to enqueue: ");
                scanf("%d", &d);
                enqueue(&q1, &q2, d);
                break;
            case 6:
                d = dequeue(&q1);
                printf("Element %d was dequeued\n", d);
                break;
            case 7:
                dispQueue(q1);
                break;
            case 8:
```

```
                return 0;
            default:
                printf("Invalid input\n");
        }
    }

    return 0;
}
```

Output:

```
djprweu5.nx0 --dbgExe=C:\\msys64\\ucrt64\\bin\\gdb.exe --interpreter=mi
Enter your choice:
1. Push to Stack
2. Pop from Stack
3. Peek Stack
4. Display Stack
5. Enqueue
6. Dequeue
7. Display Queue
8. Exit
Enter your choice: 1
Enter data to push: 10
Enter your choice: 1
Enter data to push: 5
Enter your choice: 1
Enter data to push: 6
Enter your choice: 4
Stack elements: 6 5 10
Enter your choice: 3
Top element: 6
Enter your choice: 5
Enter data to enqueue: 12
Enter your choice: 2
Popped element is: 6
Enter your choice: 5
Enter data to enqueue: 2
Enter your choice: 5
Enter data to enqueue: 6
Enter your choice: 7
Queue elements: 12 2 6
Enter your choice: 6
Element 12 was dequeued
Enter your choice: 7
Queue elements: 2 6
Enter your choice:
```

6b)
```
#include <stdio.h>
#include <stdlib.h>

struct Node
{
    int data;
    struct Node* next;
};

struct Node* createNode (int data)
{
   struct Node* newnode = (struct Node*) malloc(sizeof(struct node))
   newNode -> data = data;
   newNode -> next = NULL;
   return newNode;
}

void push (struct Node ** top, int data)
{
   struct Node* newNode = createNode (data);
   newNode -> next = *top
   * top = newNode;
   ~~print~~
}
int pop ( struct Node ** top)
{
   if (*top == NULL)
   { printf(" Stack underflow")
     ~~return~~
   else
   {
     ~~struct Node* temp = *top;~~
     ~~*top = temp -> next;~~
     ~~tet~~ int popped = temp -> data;
        * top = temp -> next;
       free(temp)
}  }
     return popped;
```

```
int init peek (struct Node* top)
  {
     if (top == NULL)
     {
        printf("Stack is Empty");
     } return 1;
     else
     {
        return top -> data;
     }

void dispStack (struct Node * top)
{
   if (top == NULL)
    {
       printf("Stack is empty");
       return 0;
    }
   else
   printf("Stack elements: \n");
   struct Node* temp = top;
   while (temp != NULL)
   {
     printf("%d", temp -> data);
     temp = temp -> next;
   }
   printf("\n");
```

PAGE :
DATE : / /

```
void enqueue( struct Node **f , struct Node **r, int data)
{
    struct Node* newNode = creatNode(data);
    if (*r ==NULL)
     {
        *f = *r = newNode;
        return;
     }
    *r -> next = newNode;
    *r -> newNode;
}

int dequeue (struct Node **f)
{
   if (*f = NULL)
    {
       printf("Underflow\n");
       return -1;
    }
  else
   {
      struct Node *temp = *f;
      int dequeued = temp -> data;
      *f = temp -> next;

      free(temp);
      return dequeued;
   }
}
```

```
void displayQueue (struct node *f)
{
    if (f == NULL)
    {
        printf("Queue is Empty\n");
        return;
    }
    printf("Queue elements: ");
    struct Node * temp = f
    while (temp != NULL)
    {
        printf("%d, ", temp->data);
        temp = temp->next;
    }
    printf("\n");
}
int main()
{
    struct Node* stack= NULL;
    printf("stack Operations:\n");
    struct Node* f = NULL, *r = NULL;
    int c, d;
    printf("Enter 1. Push to Stack\n 2. Pop Stack\n 3. Peep Stack
        4. Display Stack\n 5. Enqueue\n 6. Dequeue\n
        7. Display Queue\n 8. Exit.");

    do
    while (1)
    {
        printf("Enter your choice");
        scanf("%d", &c);
```

```
switch (c)
{
    case 1:
        printf("Enter data to push");
        scanf("%d", &d);
        push (&stackt, d);
        break;

    case 2:
        d = pop (&stackt);
        printf("Popped element is %d\n", d);
        break;
    case 3:
        d = peek (stackt);
        printf("top element: %d\n", d);
        break;
    case 4:
        displStack (stackt);

    case 5:
        printf("Enter data to enqueue");
        scanf("%d", &d);
        enqueue (&Qf, &Qr, d);
        break;
    case 6:
        d = dequeue (&Qf);
        printf("element %d was dequeued", &d);
        break;
    case 7:
        displQueue (Qf);
        break;
    case 8:
        return 0;
}
default: printf("Invalid Input");
```

Output:
Enter choice:
1. Push to stack
2. Pop stack
3. Peek Stack
4. display stack
5. Enqueue
6. Dequeue
7. Display Queue
8. Exit

o/p: Enter Choice 1
Enter data: 1
Enter Choice 1
Enter data: 5
Enter choice 4
Stack elements: 5 1
Enter Choice 2
popped