```
/*Program 1: write a program to show stack operation using array.*/
#include<stdio.h>
#include<stdlib.h>
#define MAX 10
#define TRUE 1
int stack[MAX];
int top = -1;
void push();
int pop();
void display();
void empty();
int peek();
int search(const int);
void push() {
    printf("\nEnter val: ");
    if (top == MAX - 1) {
        printf("\nStack Full, Cant push!");
        return;
    }
    ++top;
    scanf("%d", &stack[top]);
    display();
    return;
}
void display() {
    int i = 0;
    if (top == -1) {
        printf("\nEmpty, nothing to display");
        return;
    }
    printf("\n| ");
    for (i = top; i >= 0; --i)
        printf("%d ", stack[i]);
    printf("|");
    return;
}
int pop() {
    int r;
    if (top == -1) {
        printf("\nStack empty");
        return -1;
    }
    r = stack[top];
    --top;
    display();
    return r;
}
void empty() {
    int i = 0;
    while (top != -1) {
        printf("%d ", stack[top]);
        --top;
    }
    printf("\nStack empty now!");
    return;
}
int peek() {
//return -1 if empty
    if (top == -1) {
        printf("\nEmpty");
        return -1;
    } else
        printf("\nTop most : %d", stack[top]);
    return stack[top];
}
int search(const int n) {
    int i;
    for (i = top; i >= 0; --i) {
        if (stack[i] == n) {
            printf("\nfound at %d index from bottom: ", i);
            return MAX - (i + 1);
        }
    }
    printf("\nNot found");
    return -1;
}
int main() {
    int ch, tmp;
    while (TRUE) {
        printf(

"\n0.Push\n1.Pop\n2.Display\n3.Peek\n4.Empty\n5.Search\n6.Exit\nEnter your
choice");
        scanf("%d", &ch);
        switch (ch) {
        case 0:
            push();
            break;
        case 1:
            pop();
            break;
        case 2:
            display();
            break;
        case 3:
            peek();
            break;
        case 4:
            empty();
            break;
        case 5:
            printf("\nEnter no to be searched");
            scanf("%d", &tmp);
            search(tmp);
            break;
        case 6:
            exit(0);
            break;
        default:
            printf("\nWrong Choice. Retry...");
            break;
        }
    }
    return 0;
}
```

Output:

```
0.Push
1.Pop
2.Display
3.Peek
4.Empty
5.Search
6.Exit
Enter your choice0

Enter val: 65

| 65 |

| 65 |
0.Push
1.Pop
2.Display
3.Peek
4.Empty
5.Search
6.Exit
Enter your choice
0

Enter val: 45

| 45 65 |
```

```
| 48 55 98 45 65 |

0.Push
1.Pop
2.Display
3.Peek
4.Empty
5.Search
6.Exit
Enter your choice1

| 55 98 45 65 |

| 98 45 65 |
0.Push
1.Pop
2.Display
3.Peek
4.Empty
5.Search
6.Exit
Enter your choice3

Top most : 98
```

```
| 98 45 65 |

0.Push
1.Pop
2.Display
3.Peek
4.Empty
5.Search
6.Exit
Enter your choice5

Enter no to be searched45

found at 1 index from
bottom:
```

```c
/*Program 2: implement stack using linked list*/
#include<stdio.h>
#include<malloc.h>
#include<stddef.h>
#include<stdlib.h>
#define TRUE 1
typedef struct stack node;
struct stack {
    int val;
    node *next;
}*top = NULL, *tmp;

void push();
int pop();
void display();
void empty();

void push() {
    int val;
    printf("\nNode value:");
    scanf("%d", &val);
    tmp = (node *) malloc(sizeof(node));
    tmp->val = val;
    tmp->next = (top == NULL ? NULL : top);
    top = tmp;
    display();
    return;
```

```
}

void display() {
    tmp = top;
    printf("\n");
    if (tmp != NULL) {
        do {
            printf("%d ", tmp->val);
            tmp = tmp->next;
        } while (tmp != NULL);
        printf("END");
    } else
        printf("NULL");
    return;
}

int pop() {
    int r;
    if (top == NULL) {
        printf("\nStack empty!");
        return -1;
    }
    tmp = top;
    r = top->val;
    top = top->next;
    free(tmp);
    display();
    return r;
}

void empty() {
    display();
    while (top != NULL) {
        tmp = top;
        top = top->next;
        free(tmp);
    }
    printf("\nStack empty now");
}

int main() {
    int ch;
    while (TRUE) {
        printf(
                "\n0.Push\n1.Pop\n2.Display\n3.Empty\n4.Exit\nEnter your
choice");
        scanf("%d", &ch);
        switch (ch) {
        case 0:
            push();
            break;
        case 1:
            pop();
            break;
        case 2:
            display();
            break;
        case 3:
            empty();
            break;
        case 4:
            exit(0);
            break;
        default:
            printf("\nWrong Choice. Retry...");
            break;
        }
    }
    return 0;
}
```

Output:

```
0.Push
1.Pop
2.Display
3.Empty
4.Exit
Enter your choice0

Node value:65

65 END
0.Push
1.Pop
2.Display
3.Empty
4.Exit
Enter your choice0

Node value:45

45 65 END
```

```
97 23 78 45 65 END
0.Push
1.Pop
2.Display
3.Empty
4.Exit
Enter your choice1

23 78 45 65 END
0.Push
1.Pop
2.Display
3.Empty
4.Exit
Enter your choice1

78 45 65 END
```

```
78 45 65 END
0.Push
1.Pop
2.Display
3.Empty
4.Exit
Enter your choice2

78 45 65 END
0.Push
1.Pop
2.Display
3.Empty
4.Exit
Enter your choice3

78 45 65 END
Stack empty now
```

```
/*Program 3: Sort the element of a given stack by only using another stack
and push, pop operation.*/
#include<stdio.h>
#include<malloc.h>
#include<stddef.h>
#include<stdlib.h>
#define TRUE 1
typedef struct stack node;
struct stack {
    int val;
    node *next;
}*top = NULL, *tmp, *top2 = NULL;

void push(node **, int);
int pop(node **);
void display(node **);
void sort();

void push(node** head, int val) {
    tmp = (node *) malloc(sizeof(node));
    tmp->val = val;
    tmp->next = (*head == NULL ? NULL : *head);
    *head = tmp;
    return;
}

void display(node** head) {
    node *top = *head;
    tmp = top;
    printf("\n");
    if (tmp != NULL) {
        do {
            printf("%d ", tmp->val);
            tmp = tmp->next;
        } while (tmp != NULL);
        printf("END");
    } else
        printf("NULL");
    return;
}

int pop(node **head) {
    int r;
    if (*head == NULL) {
        return -1; //if empty
    }
    tmp = *head;
    r = (*head)->val;
    *head = (*head)->next;
    free(tmp);
    return r;
}

void sort() {
    int t;
    while (top) {
        while (top && (!top2 || (top2->val <= top->val))) {
            push(&top2, pop(&top));
        }
        t = pop(&top);
        if (t < 0) //if top is empty
            break;
        else {
            while (top2)
                push(&top, pop(&top2));
            push(&top, t);
        }
    }
    while (top2)
        push(&top, pop(&top2));
}

int main() {
    int ch, tmp;
    while (TRUE) {
        printf("\n0.Push\n1.Pop\n2.Display\n4.Sort\n5.Exit\nEnter your
choice");
        scanf("%d", &ch);
        switch (ch) {
        case 0:
            do {
                printf("\nvalue to be pushed: ");
                scanf("%d", &tmp);
                push(&top, tmp);
                printf("\nmore(1/0)");
                scanf("%d", &ch);
            } while (ch);
            break;
        case 1:
            pop(&top);
            break;
        case 2:
            display(&top);
            break;
        case 4:
            sort(&top, &top2);
            break;
        case 5:
            exit(0);
            break;
        default:
            printf("\nWrong Choice. Retry...");
            break;
        }
    }
    return 0;
}
```

Output:

```
0.Push
1.Pop
2.Display
4.Sort
5.Exit
Enter your choice0

value to be pushed: 65

more(1/0)1

value to be pushed: 87

more(1/0)1

value to be pushed: 36
```

```

more(1/0)1

value to be pushed: 55

more(1/0)0

0.Push
1.Pop
2.Display
4.Sort
5.Exit
Enter your choice4
```

```
0.Push
1.Pop
2.Display
4.Sort
5.Exit
Enter your choice2

36 55 65 87 END
0.Push
1.Pop
2.Display
4.Sort
5.Exit
Enter your choice5
```

```
/*Program 4: How can you implement two stacks in a single array, where no
stack overflows until no space left in the entire array space?*/
#include<stdio.h>
#include<malloc.h>
#include<stddef.h>
#include<stdlib.h>
#define TRUE 1
#define MAX 5
int stack[MAX];

int top1, top2;
top1 = -1;
top2 = MAX;
void push(int *top, int val) {
    if (top1 > top2 - 2) {
        printf("\nNo space can't push the last given value!");
        return;
    }
    if ((top == &top1) && (top1 < MAX - 1))
        ++(*top);
    else if ((top == &top2) && (top2 > 0))
```

```
        --(*top);
    else {
        printf("\nNo space can't push the last given value!");
        return;
    }
    stack[*top] = val;
    return;
}
int pop(int *top) {
    int r;
    if (top == &top1) {
        if (top1 < 0)
            return -1;
        r = stack[*top];
        --top1;
    } else if (top == &top2) {
        if (top2 > MAX - 1)
            return -1;
        r = stack[*top];
        ++top2;
    } else
        return -1;
    return r;
}
void display() {
    int i;
    printf("\nStack1 :");
    for (i = 0; i <= top1; i++)
        printf("%d ", stack[i]);
    printf("\nStack2: ");
    for (i = MAX - 1; i >= top2; --i)
        printf("%d ", stack[i]);
}
int main() {
    int ch, n, c;
    while (TRUE) {
        printf(
                "\nEnter your choice: \n1.Push in stack1\n2.Push in stack
2\n3.Pop from stack1\n4.Pop from stack 2\n5.Display\n6.Exit\n...:");
        scanf("%d", &ch);
        c = 1;
        switch (ch) {
        case 1:
            while (c) {
                printf("\nenter val:");
                scanf("%d", &n);
                push(&top1, n);
                printf("\nmore(1/0)..:");
                scanf("%d", &c);
            }
            break;
        case 2:
            while (c) {
                printf("\nenter val:");
                scanf("%d", &n);
                push(&top2, n);
                printf("\nmore(1/0)..:");
                scanf("%d", &c);
            }
            break;
        case 3:
            n = pop(&top1);
            if (!(n + 1)) {
                printf("\nAlready empty!");
            } else
                printf("\nPopped val: %d", n);
            break;
        case 4:
            n = pop(&top2);
            if (!(n + 1)) {
                printf("\nAlready empty!");
            } else
                printf("\nPopped val: %d", n);
            break;
        case 5:
            display();
            break;
        case 6:
            exit(0);
            break;
        default:
            printf("\nWrong choice ! Retry...");
            break;
        }
    }
}
```

Output:

```
Enter your choice:
1.Push in stack1
2.Push in stack 2
3.Pop from stack1
4.Pop from stack 2
5.Display
6.Exit
...:1

enter val:65

more(1/0)..:1

enter val:98

more(1/0)..:0

Enter your choice:
1.Push in stack1
2.Push in stack 2
3.Pop from stack1
4.Pop from stack 2
5.Display
6.Exit
...:2
```

```
enter val:84

more(1/0)..:1

enter val:97

more(1/0)..:0

Enter your choice:
1.Push in stack1
2.Push in stack 2
3.Pop from stack1
4.Pop from stack 2
5.Display
6.Exit
...:3

Popped val: 98
Enter your choice:
1.Push in stack1
2.Push in stack 2
3.Pop from stack1
4.Pop from stack 2
5.Display
6.Exit
...:4
```

```
Popped val: 97
Enter your choice:
1.Push in stack1
2.Push in stack 2
3.Pop from stack1
4.Pop from stack 2
5.Display
6.Exit
...:5

Stack1 :65
Stack2: 84
Enter your choice:
1.Push in stack1
2.Push in stack 2
3.Pop from stack1
4.Pop from stack 2
5.Display
6.Exit
...:
```

```
/*Program 5: Design a stack using queue*/
#include<stdio.h>
#include<malloc.h>
#include<stddef.h>
#include<stdlib.h>
#define TRUE 1
typedef struct node node;
typedef struct queue queue;
struct node {
    int val;
    node *next;
}*top = NULL, *tmp;
struct queue {
    node* front, *rear;
    int size;
} q1, q2, t;

void push();
int pop();
void display();
void init(queue *q) {
    q->front = NULL;
    q->rear = NULL;
    q->size = 0;
}
void enqueue(queue *q, int val) {
    tmp = malloc(sizeof(node));
    tmp->val = val;
    tmp->next = NULL;
    if (q->front && q->rear)
        q->rear->next = tmp;
    else {
        q->front = tmp;
    }
    q->rear = tmp;
    ++(q->size);
}
int deque(queue *q) {
    int r;
    if (!(q->front)) {
        printf("\nEmpty");
        return -1;
    }
    tmp = q->front;
    r = tmp->val;
    q->front = q->front->next;
    if (!(q->front))
        q->rear = NULL;
    free(tmp);
    --(q->size);
    return r;
}
void push() {
    int val;
    printf("\nNode value:");
    scanf("%d", &val);
    enqueue(&q1, val);
    display(q1);
    return;
}
void disp_all(node *move) {
    if (move->next == NULL) {
        printf("%d ", move->val);
        return;
    }
    disp_all(move->next);
    printf("%d ", move->val);
}

void display(queue q) {

    if (q.front == NULL) {
        printf("\nEmpty");
        return;
    }
    disp_all(q.front);
    return;
}

int pop() {
    int r;
    if ((q1.size) <= 0) {
        printf("\nEmpty");
        return -1;
    }
    while (q1.size != 1) {
        enqueue(&q2, deque(&q1));
    }
    r = deque(&q1);
    t = q1;
    q1 = q2;
    q2 = t;
    display(q1);
    return r;
}

int main() {
    int ch, tmp;
    while (TRUE) {
        printf("\n0.Push\n1.Pop\n2.Display\n3.Exit\nEnter your choice");
        scanf("%d", &ch);
        switch (ch) {
        case 0:
            push();
            break;
        case 1:
            pop();
            break;
        case 2:
            display(q1);
            break;
        case 3:
            exit(0);
            break;
        default:
            printf("\nWrong Choice. Retry...");
            break;
        }
    }
    return 0;
}
```

Output:

```
0.Push
1.Pop
2.Display
3.Exit
Enter your choice0

Node value:65
65
0.Push
1.Pop
2.Display
3.Exit
Enter your choice0

Node value:74
74 65
0.Push
1.Pop
2.Display
3.Exit
Enter your choice0
```

```
Node value:78
78 74 65
0.Push
1.Pop
2.Display
3.Exit
Enter your choice0

Node value:43
43 78 74 65
0.Push
1.Pop
2.Display
3.Exit
Enter your choice1
78 74 65
```

```
0.Push
1.Pop
2.Display
3.Exit
Enter your choice2
78 74 65

0.Push
1.Pop
2.Display
3.Exit
Enter your choice
```

```
/*Program 6: implement a doubly linked list using Stack (As much you
required)*/
#include<stdio.h>
#include<malloc.h>
#include<stddef.h>
#include<stdlib.h>
#define TRUE 1
typedef struct node node;
typedef struct stack stack;
struct node{
    int val;
    node *next;
}*tmp;
struct stack{
    node *top;
}s,s2;
//Stack ADT Operations<--------------
void init_stack(stack *);
void push(stack *,int);
int pop(stack *);
void disp_stack(stack *);
//---------------------------------->
//Doubly Linked List ADT Operations<--
void init_dll();
void ins_first();
void ins_mid();
void ins_last();
void display();
void del_first();
void del_mid();
void del_last();
//---------------------------------->
void init_dll(){
    init_stack(&s);
    init_stack(&s2);
}
void ins_first(){
    int val;
    printf("\nNode value:");
    scanf("%d",&val);
    push(&s,val);
    display();
return;
}
void ins_mid(){
    int after,val,tmp;
    printf("\nAfter : ");
    scanf("%d",&after);
    printf("\nValue : ");
    scanf("%d",&val);
    while(s.top){
        tmp=pop(&s);
        push(&s2,tmp);
        if(tmp==after){
            push(&s,val);
        break;
       }
    }
    while(s2.top){
        push(&s,pop(&s2));
    }
    display();
}
void ins_last(){
    int val;
    printf("\nValue : ");
    scanf("%d",&val);
    while(s.top)
        push(&s2,pop(&s));
    push(&s,val);
    while(s2.top)
        push(&s,pop(&s2));
    display();
return ;
}
void display(){
    disp_stack(&s);
}
void del_first(){
    pop(&s);
    display();
return ;
}
void del_mid(){
    int val;
    printf("\nNode value : ");
    scanf("%d",&val);
    while(s.top){
        tmp=pop(&s);
        if(tmp==val)
            break;
       push(&s2,tmp);
    }
    while(s2.top){
        push(&s,pop(&s2));
    }
    display();
return;
}
void del_last(){
    while(s.top)
        push(&s2,pop(&s));
    pop(&s2);
    while(s2.top)
        push(&s,pop(&s2));
    display();
return;
}
void init_stack(stack *s){
    s->top=NULL;
}
void push(stack *s,int val){
    tmp=(node *)malloc(sizeof(node));
    tmp->val=val;
    tmp->next=(s->top==NULL?NULL:s->top);
    s->top=tmp;
return;
}

int pop(stack *s){
    int r;
    if(s->top==NULL){
        return -1;
    }
    tmp=s->top;
    r=s->top->val;
    s->top=s->top->next;
    free(tmp);
return r;
}
void disp_stack(stack *s){
    tmp=s->top;
    printf("\n");
    if(tmp!=NULL)
    {
        do
        {
            printf("%d ",tmp->val);
            tmp=tmp->next;
        }while(tmp!=NULL);
    printf("END");
    }
    else
    printf("NULL");
return ;
}
int main(){
    int ch=1;
    init_dll();
    while(TRUE){
        printf("\n0.Insert at First\n1.Insert at middle\n2.Insert at
last\n3.Delete first node\n4.Delete middle node\n5.Delete last
node\n6.Diplay\n7.Exit\nEnter your choice: ");
        scanf("%d",&ch);
        switch(ch){
            case 0:ins_first();
```

```
                        break;
                case 1:ins_mid();
                        break;
                case 2:ins_last();
                        break;
                case 3:del_first();
                        break;
                case 4:del_mid();
                        break;
                case 5:del_last();
                        break;
                case 6:display();
                        break;
                case 7:exit(0);
                        break;
                default:printf("Wrong choice retry....");
                        break;
            }
        }
return 0;
}
```

Output:

```
0.Insert at First
1.Insert at middle
2.Insert at last
3.Delete first node
4.Delete middle node
5.Delete last node
6.Diplay
7.Exit
Enter your choice: 0

Node value:95

95 END

93 95 41 END
0.Insert at First
1.Insert at middle
2.Insert at last
3.Delete first node
4.Delete middle node
5.Delete last node
6.Diplay
7.Exit
Enter your choice: 1

After : 95

Value : 22

93 95 22 41 END
```

```
0.Insert at First
1.Insert at middle
2.Insert at last
3.Delete first node
4.Delete middle node
5.Delete last node
6.Diplay
7.Exit
Enter your choice: 1

After : 22

Value : 41

93 95 22 41 41 END

0.Insert at First
1.Insert at middle
2.Insert at last
3.Delete first node
4.Delete middle node
5.Delete last node
6.Diplay
7.Exit
Enter your choice: 1

After : 5

Value : 84
```

```
93 95 22 41 41 END
0.Insert at First
1.Insert at middle
2.Insert at last
3.Delete first node
4.Delete middle node
5.Delete last node
6.Diplay
7.Exit
Enter your choice: 5

93 95 22 41 END
0.Insert at First
1.Insert at middle
2.Insert at last
3.Delete first node
4.Delete middle node
5.Delete last node
6.Diplay
7.Exit
Enter your choice:7
```

```
/*Program 7: implement Queue using array.*/
#include<stdio.h>
#include<malloc.h>
#include<stddef.h>
#include<stdlib.h>
#define TRUE 1
#define MAX 10
typedef struct queue queue;
struct queue {
    int ar[MAX];
    int front, rear;
} q1;
int tmp;

void init(queue *);
void enqueue(queue *, int);
int deque(queue *);
void display(queue);

void init(queue *q) {
    q->front = -1;
    q->rear = -1;
}

void enqueue(queue *q, int val) {
    if (q->rear == MAX - 1) {
        printf("\nFULL");
        return;
    }
    if ((q->front < 0) || (q->rear < 0))
        q->front = 0;
    q->ar[++(q->rear)] = val;
}

int deque(queue *q) {
    int r;
    if ((q->front < 0) || (q->front > q->rear)) {
        printf("\nEmpty");
        return -1;
    }
    tmp = q->front;
    r = q->ar[(q->front)];
    ++(q->front);
    if ((q->front) >= MAX || q->front > q->rear) {
        q->rear = -1;
        q->front = -1;
    }
    return r;
}

void display(queue q) {
    tmp = q.front;
    printf("\nFront ");
    if (tmp >= 0) {
        do {
            printf("%d ", q.ar[tmp]);
            ++tmp;
        } while (tmp <= q.rear);
        printf("Rear");
    } else
        printf("NULL");
    return;
}
int main() {
    int ch, t, c;
    init(&q1);
    while (TRUE) {
        c = 1;
        printf("\n0.Enqueue\n1.Dequeue\n2.Display\n3.Exit\nEnter your
choice");
        scanf("%d", &ch);
        switch (ch) {
        case 0:
            while (c) {
                printf("\nValue : ");
                scanf("%d", &t);
                enqueue(&q1, t);
                printf("\nMore(1/0)..");
                scanf("%d", &c);
            }
            display(q1);
            break;
        case 1:
            deque(&q1);
            display(q1);
            break;
        case 2:
            display(q1);
            break;
        case 3:
            exit(0);
            break;
        default:
            printf("\nWrong Choice. Retry...");
            break;
        }
    }
    return 0;
}
```

Output:

```
0.Enqueue
1.Dequeue
2.Display
3.Exit
Enter your choice0

Value : 45

More(1/0)..1

Value : 74
More(1/0)..1

Value : 942

More(1/0)..1

Value : 33

More(1/0)..0
Front 45 74 942 33 Rear
0.Enqueue
1.Dequeue
2.Display
3.Exit
Enter your choice1

Front 74 942 33 Rear
0.Enqueue
1.Dequeue
2.Display
3.Exit
Enter your choice3
```

```
/*Program 8: implement Queue using linked list.*/
#include<stdio.h>
#include<malloc.h>
```

```
#include<stddef.h>
#include<stdlib.h>
#define TRUE 1
typedef struct node node;
typedef struct queue queue;
struct node {
    int val;
    node *next;
}*top = NULL, *tmp;

struct queue {
    node* front, *rear;
} q1;

void init(queue *);
void enqueue(queue *, int);
int deque(queue *);
void display(queue);

void init(queue *q) {
    q->front = NULL;
    q->rear = NULL;
}

void enqueue(queue *q, int val) {
    tmp = malloc(sizeof(node));
    tmp->val = val;
    tmp->next = NULL;
    if (q->front && q->rear)
        q->rear->next = tmp;
    else {
        q->front = tmp;
    }
    q->rear = tmp;
}

int deque(queue *q) {
    int r;
    if (!(q->front)) {
        printf("\nEmpty");
        return -1;
    }
    tmp = q->front;
    r = tmp->val;
    q->front = q->front->next;
    if (!(q->front))
        q->rear = NULL;
    free(tmp);
    return r;
}

void display(queue q) {
    tmp = q.front;
    printf("\nFront ");
    if (tmp != NULL) {
        do {
            printf("%d ", tmp->val);
            tmp = tmp->next;
        } while (tmp != NULL);
        printf("Rear");
    } else
```

```
        printf("NULL");
    return;
}
int main() {
    int ch, t, c;
    init(&q1);
    while (TRUE) {
        c = 1;
        printf("\n0.Enqueue\n1.Dequeue\n2.Display\n3.Exit\nEnter your
choice");
        scanf("%d", &ch);
        switch (ch) {
        case 0:
            while (c) {
                printf("\nValue : ");
                scanf("%d", &t);
                enqueue(&q1, t);
                printf("\nMore(1/0)..");
                scanf("%d", &c);
            }
            display(q1);
            break;
        case 1:
            deque(&q1);
            display(q1);
            break;
        case 2:
            display(q1);
            break;
        case 3:
            exit(0);
            break;
        default:
            printf("\nWrong Choice. Retry...");
            break;
        }
    }
    return 0;
}
```

Output:

```
0.Enqueue
1.Dequeue
2.Display
3.Exit
Enter your choice0

Value : 99

More(1/0)..1

Value : 74

More(1/0)..1
```

```
Value : 33

More(1/0)..1

Value : 46

More(1/0)..0

Front 99 74 33 46 Rear
```

```
0.Enqueue
1.Dequeue
2.Display
3.Exit
Enter your choice1

Front 74 33 46 Rear
0.Enqueue
1.Dequeue
2.Display
3.Exit
Enter your choice
```

```
/*Program 9: Given two integer sequences, one of which is the push sequence
of a stack, Please check whether the other sequence is a corresponding pop
sequence or not. */
#include<stdio.h>
#include<malloc.h>
#include<stddef.h>
#include<stdlib.h>
#define TRUE 1
#define MAX 10
typedef struct node node;
typedef struct stack stack;
struct node {
    int val;
    node *next;
}*tmp;
struct stack {
    node *top;
} s;
//Stack ADT Operations<--------------
void init_stack(stack *);
void push(stack *, int);
int pop(stack *);
void disp_stack(stack *);
int find(stack*, int);
//---------------------------------->
void init_stack(stack *s) {
    s->top = NULL;
}
void push(stack *s, int val) {
    tmp = (node *) malloc(sizeof(node));
    tmp->val = val;
    tmp->next = (s->top == NULL ? NULL : s->top);
    s->top = tmp;
    return;
}
int pop(stack *s) {
    int r;
    if (s->top == NULL) {
        return -1;
    }
    tmp = s->top;
    r = s->top->val;
    s->top = s->top->next;
    free(tmp);
    return r;
}
void disp_stack(stack *s) {
    tmp = s->top;
    printf("\n");
    if (tmp != NULL) {
        do {
            printf("%d ", tmp->val);
            tmp = tmp->next;
        } while (tmp != NULL);
        printf("END");
    } else
        printf("NULL");
    return;
}
int get_s(int ar[MAX]) {
    int i = 0, n;
    char buf[100], *p = buf;
    if (fgets(buf, sizeof buf, stdin) == NULL) {
        printf("problem getting numbers.. terminating!!");
        exit(0);
    }
    while ((i < MAX) && (sscanf(p, "%d%n", &ar[i], &n) == 1)) {
        i++;
        p += n;
    }
    return i;
}
int find(stack *s, int val) {
    node *move = s->top;
    while (move) {
        if (move->val == val)
            return 1;
        move = move->next;
    }
    return 0;
}
int chk_pop(int* ppush, int* ppop, int spush_length, int spop_length) {
    int npop;
    int i, j, l = 0;
    if (spush_length && spop_length) {
        i = 0;
        j = 0;
        for (i = 0; i < spop_length; i++) {
            npop = ppop[i];
            if (!find(&s, npop)) {
                for (j = l; j < spush_length; j++) {
                    push(&s, ppush[j]);
                    if (ppush[j] == npop) {
                        break;
                    }
                }
                l = j + 1;
            }
            if (s.top->val != npop) {
                return 0;
            }
            pop(&s);
        }
    }
    return 1;
}

int main() {
    int spush[MAX], spop[MAX], spush_size, spop_size;
    init_stack(&s);
    printf("\nEnter Push seq(space separated): ");
    spush_size = get_s(spush);
    printf("\nEnter Pop seq(space separated): ");
    spop_size = get_s(spop);
    if (chk_pop(spush, spop, spush_size, spop_size)) {
        printf("\nPossible");
        return 0;
    }
    printf("\nNot Possible");
    return 0;
}
```

Output:

```
Enter Push seq(space separated): 1 2 3 4 5

Enter Pop seq(space separated): 5 4 3 2 1

Possible


Enter Push seq(space separated): 1 2 3 4 5

Enter Pop seq(space separated): 4 5 3 2 1

Possible


Enter Push seq(space separated): 1 2 3 4 5

Enter Pop seq(space separated): 4 5 1 3 2

Not Possible
```

```
/*Program 10: implement a queue using stack (as much as required).*/
#include<stdio.h>
#include<malloc.h>
#include<stddef.h>
#include<stdlib.h>
#define TRUE 1

typedef struct node node;
typedef struct stack stack;
struct node {
    int val;
    node *next;
}*tmp;
struct stack {
    node *top;
    int size;
} s, s2;
//Stack ADT Operations<--------------
void init_stack(stack *);
void push(stack *, int);
int pop(stack *);
//---------------------------------->
//Queue ADT Operations<---------------
void init_queue();
void enqueue(int);
int deque();
void display_queue();
//----------------------------------->

void init_stack(stack *s) {
    s->top = NULL;
    s->size = 0;
}
void push(stack *s, int val) {
    tmp = (node *) malloc(sizeof(node));
    tmp->val = val;
    tmp->next = (s->top == NULL ? NULL : s->top);
    s->top = tmp;
    ++(s->size);
    return;
}

int pop(stack *s) {
    int r;
    if (s->top == NULL) {
        return -1;
    }
    tmp = s->top;
    r = s->top->val;
    s->top = s->top->next;
    --(s->size);
    free(tmp);
    return r;
}
void disp_all(node *move) {
    if (move->next == NULL) {
        printf("%d ", move->val);
        return;
    }
    disp_all(move->next);
    printf("%d ", move->val);
}

void display_queue() {
    stack q = s;
    if (q.top == NULL) {
        printf("\nEmpty");
        return;
    }
    disp_all(q.top);
    return;
}
void init_queue() {
    init_stack(&s);
    init_stack(&s2);
}

void enqueue(int val) {
    push(&s, val);
}

int deque() {
    int r;
    if (!s.top) {
        printf("\nEmpty");
        return -1;
    }
    while (s.size != 1) {
        push(&s2, pop(&s));
    }
    r = pop(&s);
    while (s2.size)
        push(&s, pop(&s2));
    return r;
}

int main() {
    int ch, t, c;
    init_queue();
    while (TRUE) {
        c = 1;
        printf("\n0.Enqueue\n1.Dequeue\n2.Display\n3.Exit\nEnter your
choice");
        scanf("%d", &ch);
        switch (ch) {
        case 0:
            while (c) {
                printf("\nValue : ");
                scanf("%d", &t);
                enqueue(t);
                printf("\nMore(1/0)..");
                scanf("%d", &c);
            }
            display_queue();
            break;
        case 1:
            deque();
            display_queue();
            break;
        case 2:
            display_queue();
            break;
        case 3:
            exit(0);
            break;
        default:
            printf("\nWrong Choice. Retry...");
            break;
        }
    }
    return 0;
}
```

Output:

```
0.Enqueue
1.Dequeue
2.Display
3.Exit
Enter your choice0

Value : 11

More(1/0)..1

Value : 42

More(1/0)..1
```

```
Value : 33

More(1/0)..0
11 42 33
0.Enqueue
1.Dequeue
2.Display
3.Exit
Enter your choice1
```

```
42 33
0.Enqueue
1.Dequeue
2.Display
3.Exit
Enter your choice
```