Problem-1

```
#include<iostream>
#include<string>
using namespace std;
#define MAXSIZE 100
#define OVERFLOW -2
#define OK       1
#define ERROR    0
#define TRUE    1
#define FALSE    0
typedef int Status;
typedef int QElemType;
class  SqQueue
{
protected:
    QElemType  *base;
    int   front;
    int   rear;
public:
    int n;
    SqQueue();
    ~SqQueue();
    Status EnQueue(QElemType e);
    Status DeQueue(QElemType &e);
    Status QueueEmpty();
    Status QueueFull();
    void  PrintQueue();
};
SqQueue::SqQueue()
{
    base = new QElemType[MAXSIZE];
    if (!base) exit(OVERFLOW);
    front = rear = 0;
}
SqQueue::~SqQueue()
{
    delete base;
}
Status SqQueue::EnQueue(QElemType e)
{

    if (QueueFull())
```

```
    {
        cout << "Queue is Full" << endl;
        return ERROR;
    }
    base[rear] = e;
    rear = (rear + 1) % (n + 1);
    return OK;
}
Status SqQueue::DeQueue(QElemType &e)
{
    if (QueueEmpty())
    {
        cout << "Queue is Empty" << endl;
        return ERROR;
    }
    e = base[front];
    cout << e << endl;
    front = (front + 1) % (n + 1);
    return OK;
}
Status SqQueue::QueueEmpty()
{
    if (front == rear) return TRUE;
    return FALSE;
}
Status SqQueue::QueueFull()
{
    if ((rear + 1) % (n + 1) == front)
        return TRUE;
    return FALSE;
}
void SqQueue::PrintQueue()
{
    while (front % (n + 1) != rear)
    {
        cout << base[front % (n + 1)] << " ";
        front = (front + 1) % (n + 1);
    }
    cout << endl;
}
int main()
{
    SqQueue  Q;
    QElemType e;
```

```
    cin >> Q.n;
    string s;
    int x;
    int y[100] = { 0 };
    string vis[100];
    int i = 0;
    while (1)
    {
        cin >> s;
        if (s == "dequeue")
            vis[i] = "dequeue";

        else if (s == "enqueue")
        {
            cin >> x;
            vis[i] = "enqueue";
            y[i] = x;
        }
        else
        {
            vis[i] = "quit";
            break;
        }
        i++;
    }
    int k = 0;
    while (vis[k] != "quit")
    {
        if (vis[k] == "dequeue")
            Q.DeQueue(e);
        else
            Q.EnQueue(y[k]);
        k++;

    }
    Q.PrintQueue();
    return 0;
}
```

Problem-2

```
#include<iostream>
#include<string>
using namespace std;
#define MAXSIZE 100
#define OVERFLOW -2
#define OK       1
#define ERROR    0
#define TRUE    1
#define FALSE    0
typedef int Status;
typedef int QElemType;
class QNode
{
public:
    QElemType  data;
    QNode     *next;
};
class LinkQueue :public QNode
{
protected:
    QNode  *front;
    QNode  *rear;
    int count;
public:
    int n;
    LinkQueue();
    ~LinkQueue();
    Status EnQueue(QElemType e);
    Status DeQueue(QElemType &e);
    Status QueueEmpty();
    void  PrintQueue();
};
LinkQueue::LinkQueue()
{
    front = rear = new QNode;
    if (!front) exit(OVERFLOW);
    count = 0;
    front->next = NULL;
}
LinkQueue::~LinkQueue()
{
```

```
    while (front)
    {
        rear = front->next;
        delete front;
        front = rear;
    }
}
Status LinkQueue::EnQueue(QElemType e)
{
    QNode *p = new QNode;
    if (count >= n)
    {
        cout << "Queue is Full" << endl;
        return ERROR;
    }
    if (!p) exit(OVERFLOW);
    p->data = e;  p->next = NULL;
    rear->next = p;
    rear = p;
    count++;
    return OK;
}
Status LinkQueue::DeQueue(QElemType &e)
{
    if (QueueEmpty())
    {
        cout << "Queue is Empty" << endl;
        return ERROR;
    }
    QNode *p;
    p = front->next;
    e = p->data;
    cout << e << endl;
    front->next = p->next;
    if (rear == p) rear = front;
    delete p;
    count--;
    return OK;
}
Status LinkQueue::QueueEmpty()
{
    if (front == rear) return TRUE;
    return FALSE;
}
```

```
void LinkQueue::PrintQueue()
{
    while (front->next)
    {
        cout << front->next->data << " ";
        front = front->next;
    }
    cout << endl;
}
int main()
{
    LinkQueue  Q;
    QElemType e;
    cin >> Q.n;
    string s;
    int x;
    int y[100] = { 0 };
    string vis[100];
    int i = 0;
    while (1)
    {
        cin >> s;
        if (s == "dequeue")
            vis[i] = "dequeue";
        else if (s == "enqueue")
        {
            cin >> x;
            vis[i] = "enqueue";
            y[i] = x;
        }
        else
        {
            vis[i] = "quit";
            break;
        }
        i++;
    }
    int k = 0;
    while (vis[k] != "quit")
    {
        if (vis[k] == "dequeue")
            Q.DeQueue(e);
        else
            Q.EnQueue(y[k]);
```

```
        k++;

    }
    Q.PrintQueue();
    return 0;
}
```