Harsh_Kumar_MAN_106_Assignment_3

- Answer 1:

```cpp
#include<bits/stdc++.h>
using namespace std;

#define MAX 50

class QUEUE{
    int q_fornt;
    int q_rear;
    int queue[MAX];
    int size;
public:
    QUEUE();
    bool underflow();
    bool overflow();
    void add(int item);
    int remove();
    int count();
};

QUEUE::QUEUE(){
    queue[MAX] = {0};
    q_fornt = MAX-1;
    q_rear = MAX-1;
    size = 0;
}

bool QUEUE::underflow(){
    if (size==0) return true;
    else return false;
}

bool QUEUE::overflow(){
    if (q_rear==-1) return true;
    else return false;
}

```

```
void QUEUE::add(int item){
    if (overflow()) {
        cout << "Queue is full\nRemove some values first\n";
        return;
    }
    else queue[q_rear--] = item, size++;
}

int QUEUE::remove(){
    if (underflow()) {
        cout << "Queue is empty\nAdd some values first\n";
        return -1;
    }
    else {
        size--;
        return (queue[q_fornt--]);
    }
}

int QUEUE::count(){
    return size;
}

int main(){
    QUEUE q;
    for(int i=0; i<10; i++) q.add(i+1);

    cout << q.remove() << endl;
    for(int i=0; i<9; i++) cout << q.remove() << endl ;
    cout << q.remove();
}
```

- Output :

```
1
2
3
4
5
6
7
8
9
10
Queue is empty
Add some values first
-1
```

- Answer 2_1:

```cpp
#include<bits/stdc++.h>
using namespace std;

#define MAX 50

class QUEUE{
    int q_fornt;
    int q_rear;
    char queue[MAX];
    int size;
public:
    QUEUE();
    bool underflow();
    bool overflow();
    void add(char item);
    char remove();
    int count();
};

QUEUE::QUEUE(){
    queue[MAX] = {0};
    q_fornt = MAX-1;
    q_rear = MAX-1;
    size = 0;
}

bool QUEUE::underflow(){
    if (size==0) return true;
    else return false;
}

bool QUEUE::overflow(){
    if (q_rear==-1) return true;
    else return false;
}

```

```cpp
void QUEUE::add(char item){
    if (overflow()) {
        cout << "Queue is full\nRemove some values first\n";
        return;
    }
    else queue[q_rear--] = item, size++;
}

char QUEUE::remove(){
    if (underflow()) {
        cout << "Queue is empty\nAdd some values first\n";
        return '-';
    }
    else {
        size--;
        return (queue[q_fornt--]);
    }
}

int QUEUE::count(){
    return size;
}

int main(){
    QUEUE q;
    for(int i=66; i<76; i++) q.add(char(i+1));

    cout << q.remove() << endl;
    for(int i=0; i<9; i++) cout << q.remove() << endl ;
    q.remove();
}
```

- Output :

```
C
D
E
F
G
H
I
J
K
L
Queue is empty
Add some values first
```

- Answer 2_2 :

```cpp
#include<iostream>
#include<string>

using namespace std;

#define MAX 50

class QUEUE{
    int q_fornt;
    int q_rear;
    string queue[MAX];
    int size;
public:
    QUEUE();
    bool underflow();
    bool overflow();
    void add(string item);
    string remove();
    int count();
};

QUEUE::QUEUE(){
    queue[MAX] = {0};
    q_fornt = MAX-1;
    q_rear = MAX-1;
    size = 0;
}

bool QUEUE::underflow(){
    if (size==0) return true;
    else return false;
}

bool QUEUE::overflow(){
    if (q_rear==-1) return true;
    else return false;
```

```cpp
void QUEUE::add(string item){
    if (overflow()) {
        cout << "Queue is full\nRemove some values first\n";
        return;
    }
    else queue[q_rear--] = item, size++;
}

string QUEUE::remove(){
    if (underflow()) {
        cout << "Queue is empty\nAdd some values first\n";
        return "NA";
    }
    else {
        size--;
        return (queue[q_fornt--]);
    }
}

int QUEUE::count(){
    return size;
}

int main(){
    QUEUE q;
    string s = "Hello World";
    for(int i=49; i<58; i++){
        s.push_back(char(i));
        q.add(s);
    }
    cout << q.remove() << endl;

    for(int i=0; i<9; i++) cout << q.remove() << endl ;
    q.remove();

}
```

- Output :

```
Hello World1
Hello World12
Hello World123
Hello World1234
Hello World12345
Hello World123456
Hello World1234567
Hello World12345678
Hello World123456789
Queue is empty
Add some values first
NA
```

- Answer 3 :

```cpp
#include<bits/stdc++.h>
using namespace std;

#define MAX 10

class QUEUE{
    int q_fornt;
    int q_rear;
    int queue[MAX];
    int size;
public:
    QUEUE();
    bool underflow();
    bool overflow();
    void add(int item);
    int remove();
    int count();
};

QUEUE::QUEUE(){
    queue[MAX] = {0};
    q_fornt = MAX-1;
    q_rear = MAX-1;
    size = 0;
}

bool QUEUE::underflow(){
    if (size==0) return true;
    else return false;
}

bool QUEUE::overflow(){
    if (size==MAX) return true;
    else return false;
}

```

```cpp
void QUEUE::add(int item){
    if (overflow()) {
        cout << "Queue is full\nRemove some values first\n";
        return;
    }
    else queue[(q_rear--)%MAX] = item, size++;
}

int QUEUE::remove(){
    if (underflow()) {
        cout << "Queue is empty\nAdd some values first\n";
        return -1;
    }
    else {
        size--;
        return (queue[(q_fornt--)%MAX]);
    }
}

int QUEUE::count(){
    return size;
}

int main(){
    QUEUE q;
    for(int i=0; i<11; i++) q.add(i+1);

    cout << q.remove() << endl; // As sopn as your remove a space is ceated
    q.add(42); // Added due to circular queue
    for(int i=0; i<10; i++) cout << q.remove() << endl ;

}
```

- Output :

```
Queue is full
Remove some values first
1
2
3
4
5
6
7
8
9
10
42
```

- Answer 4 :

```cpp
#include<bits/stdc++.h>
using namespace std;

#define MAX 20

class QUEUE{
    int q_front;
    int q_rear;
    int queue[MAX];
    int size;
public:
    QUEUE();
    bool underflow();
    bool overflow();
    void add_front(int item);
    void add_back(int item);
    int remove_front();
    int remove_back();
    int count();
};

QUEUE::QUEUE(){
    queue[MAX] = {0};
    q_front = 0;
    q_rear = 1;
    size = 0;
}

bool QUEUE::underflow(){
    if (q_front==0 || q_rear==1) return true;
    else return false;
}

bool QUEUE::overflow(){
    if (q_front==q_rear) return true;
    else return false;
```

```
void QUEUE::add_front(int item){
    if (overflow()) {
        cout << "Queue is full\nRemove some values first\n";
        return;
    }
    else {
        q_front-= 1;
        if (q_front==-1) q_front = MAX-1;
        queue[(q_front)%MAX] = item, size++;
    }
}

void QUEUE::add_back(int item){
    if (overflow()) {
        cout << "Queue is full\nRemove some values first\n";
        return;
    }
    else {
        q_rear+=1;
        queue[(q_rear)%MAX] = item, size++;
    }
}

int QUEUE::remove_front(){
    if (underflow()) {
        cout << "Queue is empty\nAdd some values first\n";
        return -1;
    }
    else {
        size--;
        return (queue[(q_front++)%MAX]);
    }
}
```

```
int QUEUE::remove_back(){
    if (underflow()) {
        cout << "Queue is empty\nAdd some values first\n";
        return -1;
    }
    else {
        size--;
        return (queue[(q_rear--)%MAX]);
    }
}

int QUEUE::count(){
    return size;
}

int main(){
    QUEUE q;
    for(int i=0; i<5; i++) q.add_front(i+1);
    for(int i=10; i<15; i++) q.add_back(i);
    cout << "Removing from back:\n";
    for(int i=0; i<4; i++) cout << q.remove_back() << endl ;
    cout << "Removing from front:\n";
    for(int i=0; i<5; i++) cout << q.remove_front() << endl ;
}
```

- Output :