QUEUE

1. IMPLEMENTATION OF QUEUE USING ARRAY

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


class Queue {
public:
        int front, rear, size;
        unsigned capacity;
        int* array;
};


Queue* createQueue(unsigned capacity)
{
        Queue* queue = new Queue();
        queue->capacity = capacity;
        queue->front = queue->size = 0;


        queue->rear = capacity - 1;
        queue->array = new int[(
                queue->capacity * sizeof(int))];
        return queue;
}
```

```
int isFull(Queue* queue)
{
        return (queue->size == queue->capacity);
}


int isEmpty(Queue* queue)
{
        return (queue->size == 0);
}


void enqueue(Queue* queue, int item)
{
        if (isFull(queue))
                return;
        queue->rear = (queue->rear + 1)
                                % queue->capacity;
        queue->array[queue->rear] = item;
        queue->size = queue->size + 1;
        cout << item << " enqueued to queue\n";
}


int dequeue(Queue* queue)
{
        if (isEmpty(queue))
                return INT_MIN;
        int item = queue->array[queue->front];
        queue->front = (queue->front + 1)
                                % queue->capacity;
        queue->size = queue->size - 1;
        return item;
}
```

```
int front(Queue* queue)
{
	if (isEmpty(queue))
		return INT_MIN;
	return queue->array[queue->front];
}


int rear(Queue* queue)
{
	if (isEmpty(queue))
		return INT_MIN;
	return queue->array[queue->rear];
}


int main()
{
	Queue* queue = createQueue(1000);

	enqueue(queue, 10);
	enqueue(queue, 20);
	enqueue(queue, 30);
	enqueue(queue, 40);

	cout << dequeue(queue)
		<< " dequeued from queue\n";

	cout << "Front item is "
		<< front(queue) << endl;
	cout << "Rear item is "
		<< rear(queue) << endl;

	return 0;
}
```

2. IMPLEMENTATION OF QUEUE USING LINKED LIST

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

struct QNode {
        int data;
        QNode* next;
        QNode(int d)
        {
                data = d;
                next = NULL;
        }
};

struct Queue {
        QNode *front, *rear;
        Queue()
        {
                front = rear = NULL;
        }

        void enQueue(int x)
        {


                QNode* temp = new QNode(x);


                if (rear == NULL) {
                        front = rear = temp;
                        return;
                }
                rear->next = temp;
                rear = temp;
        }


        void deQueue()
        {

                if (front == NULL)
                        return;


                QNode* temp = front;
                front = front->next;


                if (front == NULL)
                        rear = NULL;

                delete (temp);
        }
};


int main()
{

        Queue q;
        q.enQueue(10);
        q.enQueue(20);
        q.deQueue();
        q.deQueue();
        q.enQueue(30);
        q.enQueue(40);
        q.enQueue(50);
        q.deQueue();
        cout << "Queue Front : " << (q.front)->data << endl;
        cout << "Queue Rear : " << (q.rear)->data;
}
```

3. QUEUE IN C++ STL

```
#include <iostream>
#include <queue>
using namespace std;

int main()
{
   queue <int> q;
   q.push(10);
   q.push(20);
   q.push(30);


   cout << q.front() << " " << q.back() << endl;

   q.pop();

   cout << q.front() << " " << q.back() << endl;


   return 0;
}


#include <iostream>
#include <queue>
```

```
using namespace std;

int main()
{
    queue <int> q;
    q.push(10);
    q.push(20);
    q.push(30);


   while(q.empty() == false)
   {
       cout << q.front() << " " << q.back() << endl;
       q.pop();
   }


    return 0;
}
```

```
#include <iostream>
#include <queue>
using namespace std;

int main()
{
    queue <int> q;
    q.push(10);
    q.push(20);
    q.push(30);


    cout << q.size();
```

```
   return 0; }
```

4. IMPLEMENTATION OF STACK USING QUEUE

```
/* Program to implement a stack using
two queue */
#include <bits/stdc++.h>

using namespace std;

struct  Stack {

        queue<int> q1, q2;
        int curr_size;

public:
        Stack()
        {
                curr_size = 0;
        }

        void push(int x)
        {
                curr_size++;

                // Push x first in empty q2
                q2.push(x);

                // Push all the remaining
                // elements in q1 to q2.
                while (!q1.empty()) {
                        q2.push(q1.front());
                        q1.pop();
                }
                // swap the names of two queues
                queue<int> q = q1;
                q1 = q2;
                q2 = q;
        }

        void pop()
        {

                // if no elements are there in q1
                if (q1.empty())
                        return;
                q1.pop();
                curr_size--;
        }

        int top()
        {
                if (q1.empty())
                        return -1;
                return q1.front();
        }

        int size()
        {
                return curr_size;
        }
};


int main()
{
        Stack s;
        s.push(10);
        s.push(5);
        s.push(15);
        s.push(20);

        cout << "current size: " << s.size() << endl;
        cout << s.top() << endl;
        s.pop();
        cout << s.top() << endl;
        s.pop();
        cout << s.top() << endl;

        cout << "current size: " <<  s.size() << endl;
        return 0;
}
```

5. REVERSING A QUEUE

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


void Print(queue<int>& Queue)
{
        while (!Queue.empty()) {
                cout << Queue.front() << " ";
                Queue.pop();
        }
}


void reverseQueue(queue<int>& Queue)
{
        stack<int> Stack;
        while (!Queue.empty()) {
                Stack.push(Queue.front());
                Queue.pop();
        }
        while (!Stack.empty()) {
                Queue.push(Stack.top());
                Stack.pop();
        }
}


int main()
{
        queue<int> q;
        q.push(12);
        q.push(5);
        q.push(15);
        q.push(20);

        reverseQueue(q);
        Print(q);
}
```

6. GENERATE NUMBERS WITH GIVEN DIGITS

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


void printFirstN(int n)
{
   queue<string> q;
```

```
    q.push("5");
    q.push("6");

    for(int i = 0; i < n; i++)
    {
        string curr = q.front();

        cout << curr << " ";

        q.pop();

        q.push(curr + "5");
        q.push(curr + "6");
    }

}


int main()
{
         int n = 5;

         printFirstN(n);
}
```