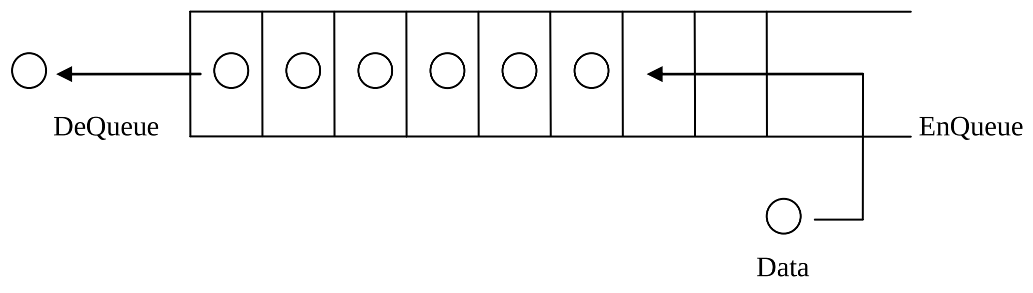
# Project Title: Implement of the class Queue

## Requirement Analysis

Queue is a kind of First-In-First-Out (FIFO) data structure, that is, the first element added to a queue will occur in the first place in the queue, the second element added to the queue will be after the first one, and so on. The element in the front will be removed from the queue before the others whenever elements dequeued. Therefore, the element added to the queue firstly must be removed before the one added later. That is the reason why it is called FIFO. There are many kinds of method to implement the functions: such as, to create sequential queue by an array, or to build linked queue by a linked list and so on.



Description Diagram of Queue

You need to design and implement a class Queue to store integer data in this project. The requirements are listed as follows:

- The size of a queue could be set while creating the queue and the default max size is ten elements.
- The elementary functions include the following (member functions of the class Queue):
  1. bool enqueue (int); Add an integer element to the queue, and show the user specific tips according to the queue size.
  2. int dequeue (); Return the first element in the queue and remove it from the queue.
  3. int peek (int) const; Return the element in the position i of the queue, without removing the element.

4. int getSize() const; Return the current size of the queue.
5. void display() const; Output and display all the elements in the current queue on the screen.

- Write a main function to test the queue above, including the requirements as follows:

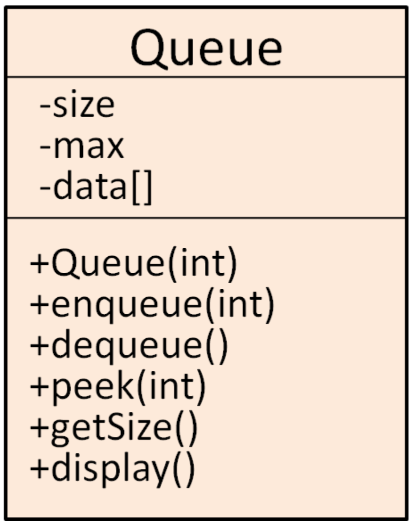
1. User inputs the queue size at first and then the program creates a queue.
2. Ask the user to input several integers to fill the queue.
3. Display the queue elements on the screen.
4. Display the queue elements after deleting the first one in the queue.

- For example, after the first running, the results are displayed on the screen as follows:

```
Please input the queue size: 5              //the inputted data is marked orange
Please input an integer to fill the queue 1
Please input an integer to fill the queue 2
Please input an integer to fill the queue 3
Please input an integer to fill the queue 4
Please input an integer to fill the queue 5
There are 5 elements in the queue. They are:
1 2 3 4 5
This is Dequeue operation: 1
There are 4 elements in the queue. They are:
2 3 4 5
```

Any output other than this one is considered wrong.

## 1. Design Description



## 2. Debug Analysis

1. For the robustness and correctness of the program, illegal inputs should be carefully taken into consideration. For instance, if an index which is out of the queue's range, the program should raise an error message instead of computing the incorrect answer.

2. Only if the header file, its actualization file and the main function have been correctly associated will the program operates smoothly. And to achieve this, 'main.cpp' should include 'queue.h' and 'queue.cpp' should be included by 'queue.h'.

## 3. Test Result

a )

```
Please input the queue size: -1
Error: incorrect queue size.
```

b )

```
Please input the queue size: 0
There is 0 element in the queue.
Error: There are no elements left to dequeue.
This is Dequeue operation: 0
There is 0 element in the queue.
```

c )

```
Please input the queue size: 1

Please input an integer to fill the queue 6
There is 1 element in the queue. It is:
6
This is Dequeue operation: 6
There is 0 element in the queue.
```

d )

```
Please input the queue size: 6

Please input an integer to fill the queue 1

Please input an integer to fill the queue 2

Please input an integer to fill the queue 3

Please input an integer to fill the queue 4

Please input an integer to fill the queue 5

Please input an integer to fill the queue 6

There are 6 elements in the queue. They are:

1 2 3 4 5 6

This is Dequeue operation: 1

There are 5 elements in the queue. They are:

2 3 4 5 6
```

e )

```
Please input the queue size: 20

Error: incorrect queue size.
```

## 4. Appendix

All the files are packed together with this report in the compressed files. However, the code as well as annotation is still provided in this report in case.

[queue.h]

```
//File name:queue.h
//Header file of queue library
#ifndef QUEUE
#define QUEUE
class Queue
{
```

```
private:
        int size,max; //Record the current size of the queue and its limit
        int data[10]; //Memorize data integers
public:
        Queue(int); //Initialize the queue
        Queue(const Queue&);//Copy constructor
        bool enqueue(int); //Add a number to the tail of the queue,
                              //returning false if failing to add
        int dequeue(); //Remove a number from the head of    the queue and return it
        int peek(int) const; //Return the element in the position i of the queue,
                              //without removing the element
        int getSize() const; //Return the current size of the queue
        void display() const; //Output and display all the elements in
                                  //the current queue on the screen
};
#include "queue.cpp"
#endif
```

```
[queue.cpp]
//file name:queue.cpp
//This file realizes the queue library.
#include <iostream>
#include <cstdlib>
using namespace std;
Queue::Queue(int num)
{
        if ((num>=0)&&(num<=10))
        {
                size=0;
                max=(10>num)?num:10;
        } else
        {
                cout<<"Error: incorrect queue size.\n";
                exit(1);
        }
} //Initialize the queue
Queue::Queue(const Queue &q)
{
        size=q.size;
        max=q.max;
        int data[10];
        for (int i=0;i<max;++i) data[i]=q.data[10];
}//Copy constructor
bool Queue::enqueue(int num) //Add a number to the tail of the queue,
                                          //returning false if failing to add
{
        if (size<max)
        {
```

```
		++size;
		data[size-1]=num;
		return true;
	} else
	{
		cout<<"Error: The queue is full.\n";
		return false;
	}
}
int Queue::dequeue() //Remove a number from the head of    the queue and return it
{
	if (size>0)
	{
		int temp=data[0],i;
		for (i=0;i<size-1;++i) data[i]=data[i+1];
		--size;
		return temp;
	} else
	{
		cout<<"Error: There are no elements left to dequeue.\n";
		return 0;
	}
}
int Queue::peek(int num) const //Return the element in the position i of the queue,
					//without removing the element
{
	if ((num>0)&&(num<max+1)) return data[num-1];
	else cout<<"Error: Queue index out of range.\n";
}
int Queue::getSize() const //Return the current size of the queue
{
	return size;
}
void Queue::display() const //Output and display all the elements in
				   //the current queue on the screen
{
	cout<<"There ";
	if (size>1) cout<<"are "; else cout<<"is ";
	cout<<size<<" element";
	if (size>1) cout<<'s';
	cout<<" in the queue.";
	if (size>1)
	{
		cout<<" They are:\n";
		int i;
		for (i=0;i<size;++i) cout<<data[i]<<' ';
		cout<<endl;
	} else if (size==1) cout<<" It is:\n"<<data[0]<<endl; else cout<<endl;
}
```

[main.cpp]

```
//File name:main.cpp
//This program tests the application of the class Queue.
#include "queue.h"
#include <iostream>
using namespace std;
int main()
{
	cout<<"Please input the queue size: ";
	int Size,i,n;
	cin>>Size; //Prompt the user to input the queue size
	Queue q(Size); //Correspondingly create a queue
	for (i=0;i<Size;++i)
	{
		cout<<"Please input an integer to fill the queue ";
		cin>>n;
		q.enqueue(n);
	} //Ask the user to input a certain number of integer(s) to fill the queue
	q.display(); //Display the queue
	cout<<"This is Dequeue operation: "<<q.dequeue()<<endl; //Delete the first element
	                                                          //in the queue
	q.display(); //Display the queue again
}
```