Name – Bhakti Bapurao Patil

Reg.no – 2020BIT064

Assignment N0 1 – Write a program to perform stack and queue

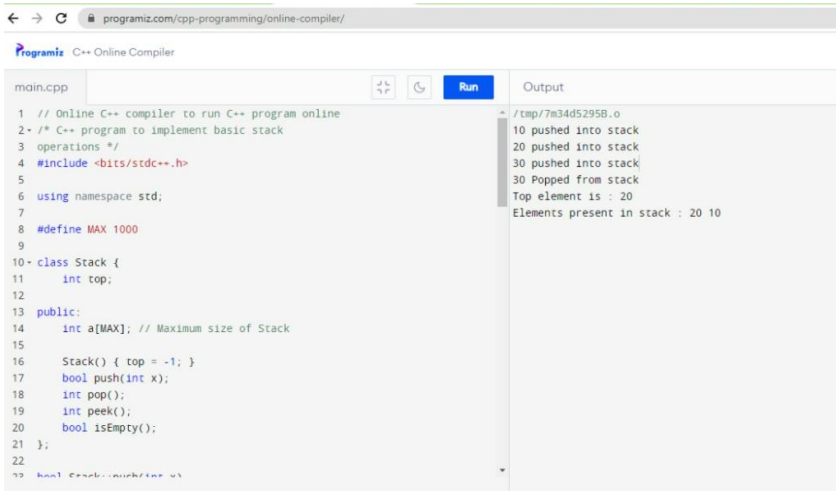
Stack

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

#define MAX 1000
class Stack {
	int top;

public:
	int a[MAX];

	Stack() { top = -1; }
	bool push(int x);
	int pop();
	int peek();
	bool isEmpty();
};

bool Stack::push(int x)
```

```
{
        if (top >= (MAX - 1)) {
                cout << "Stack Overflow";
                return false;
        }
        else {
                a[++top] = x;
                cout << x << " pushed into stack\n";
                return true;
        }
}

int Stack::pop()
{
        if (top < 0) {
                cout << "Stack Underflow";
                return 0;
        }
        else {
                int x = a[top--];
                return x;
        }
}
int Stack::peek()
{
```

```
	if (top < 0) {
		cout << "Stack is Empty";
		return 0;
	}
	else {
		int x = a[top];
		return x;
	}
}

bool Stack::isEmpty()
{
	return (top < 0);
}

int main()
{
	class Stack s;
	s.push(10);
	s.push(20);
	s.push(30);
	cout << s.pop() << " Popped from stack\n;
	cout << "Top element is : " << s.peek() << endl;

	cout <<"Elements present in stack : ";
```

```
        while(!s.isEmpty())
        {
                cout << s.peek() <<" ";
                s.pop();
        }

        return 0;
}
```



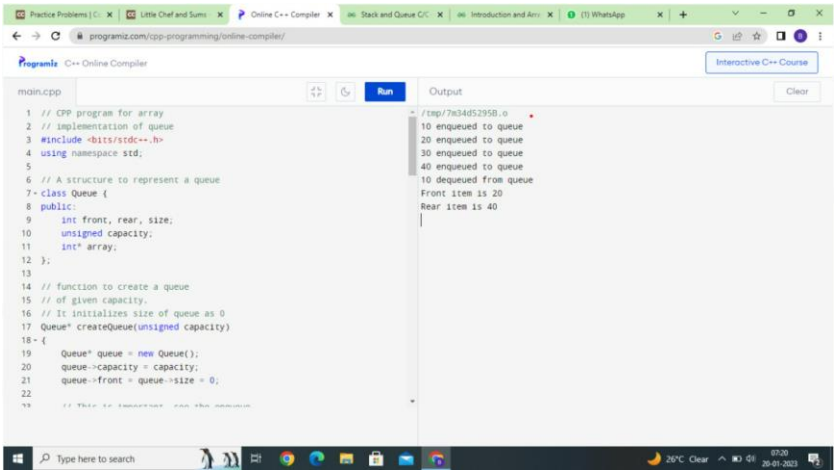
2) queue

```
#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];
        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;

}
```


Practice Problems | C++
Little Chef and Sums
Online C++ Compiler
Stack and Queue C/C++
Introduction and Arrays
(1) WhatsApp
programiz.com/cpp-programming/online-compiler/
Programiz C++ Online Compiler
Interactive C++ Course
main.cpp
Run
Output
Clear
// CPP program for array
// implementation of queue
#include <bits/stdc++.h>
using namespace std;
// A structure to represent a queue
class Queue {
public:
int front, rear, size;
unsigned capacity;
int* array;
};
// function to create a queue
// of given capacity.
// It initializes size of queue as 0
Queue* createQueue(unsigned capacity)
{
Queue* queue = new Queue();
queue->capacity = capacity;
queue->front = queue->size = 0;
/tmp/7m34d5295B.o
10 enqueued to queue
20 enqueued to queue
30 enqueued to queue
40 enqueued to queue
10 dequeued from queue
Front item is 20
Rear item is 40
Type here to search
26°C Clear
07:20
20-01-2023