Stack implementation using queue

```
# include <stdio.h>

# include <stdlib.h>

typedef struct {

    int * data ;
    int front ;
    int rear ;
    int size ;

} Queue ;

typedef struct {

    Queue* q1 ;
    Queue * q2 ;

} Mystack ;

Queue *  createQueue (int size) {

  Queue * queue = (Queue *) malloc (sizeof (Queue)) ;

  queue → data = (int *) malloc(size * sizeof (int)) ;

        queue → front = queue → rear = -1 ;

        queue → size = size ;

        return queue ;

  }

void enqueue (Queue * queue , int value ){

      if (queue → rear == -1) {

           queue → front = queue → rear = 0 ;

    }

    else {

           queue -> rear = (queue -> rear +1) % queue ->
                                                    size ;

      }
```

```
        queue -> data [queue -> rear] = value;

    }

    int dequeue (Queue* queue) {

        int value = queue -> data [queue -> front];

        if (queue -> front == queue -> rear) {

            queue -> front == queue -> rear = -1;

        }

        else {

            queue -> front = (queue -> front + 1) % queue,
                                                        size;

        }

        return value;

    }

    bool isEmpty (Queue* queue) {

        return queue -> front == -1;

    }

    MyStack* myStackCreate () {

        MyStack* stack = (MyStack*) malloc (sizeof
                                             (MyStack));

        stack -> q2 = createQueue (1000);
                  // Adjust the size ~~of~~ as needed

        stack -> q2 = createQueue(1000);

        return stack;

    }
```

```
void   myStackPush ( MyStack* obj , int x) {
        enqueue( obj -> q2, x);
  }

int  myStackTop ( MyStack* obj ) {
      if  ( isEmpty (obj -> q1 ) ) {
              return -1;      // stack is empty
      }
   while ( obj -> q2 -> front != obj -> q1 -> rear) {
          enqueue ( obj -> q2 , dequeue (obj -> q1));
   }
 int poppedValue = dequeue (obj -> q1);
 // Swap q1 and q2
 Queue* temp = obj -> q1;
 obj -> q1 = obj -> q2;
 obj -> q2 = temp;
   return  poppedValue;
}
int myStackTop (MyStack* obj ) {
      if (isEmpty (obj -> q1)) {
              return -1;    // stack is empty
  }
  while ( obj -> q1 -> front != obj -> q1 -> rear) {
      enqueue ( obj -> q2 , dequeue (obj -> q1));
  }
~~int~~
int topValue = dequeue(obj -> q1);
enqueue(obj -> q2, topValue);

// Swap q1 and q2

Queue* temp = obj -> q1;
obj -> q1 = obj -> q2;
obj -> q2 = temp;

  return topValue;

}

bool mystackEmpty(MyStack* obj){

        return isEmpty(obj -> q1);
  }

void mystackFree(MyStack* obj) {
        free(obj -> q1 -> data);
        free(obj -> q1);
        free(obj -> q2 -> data);
        free(obj -> q2);
        free(obj);
  }
```

Output

```
  [,1,2,_ _ _ _ _ _ _

null, null, null, 2, 1, false
```

[illegible]
10.01.24