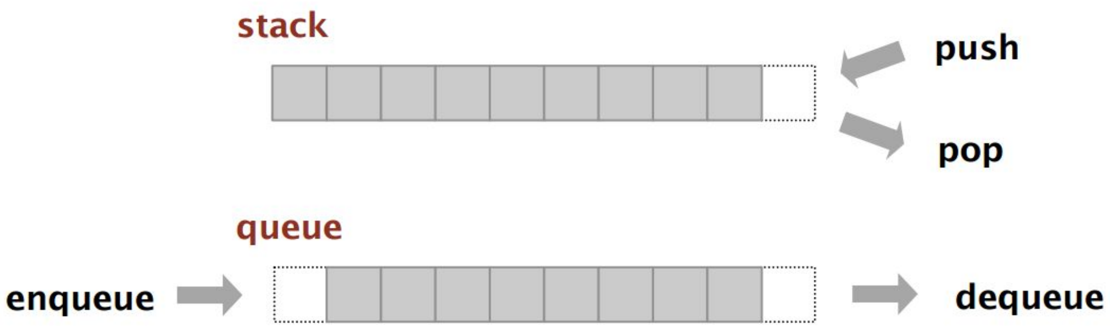
ESE 2025
Instructor: professor Taski Zourntos
Student: Vy Nguyen
Lab report: Stack and Queue

Introduction:
This assignment performs the queue-equivalent of the stacks project located from: https://github.com/takiszourntos/teaching/tree/master/lambton/2020/summer/ese2025/week_3/workspace

Discussion:
Stack and Queue operation is different from each other. While stack using the last-in-first-out (LIFO), the queue is using first-in-first-out (FIFO) operation.
The main operations of the stack are push, pop. We using push to insert an item on top the stack, and pop to remove the newest added. An array is empty when the first element that was added, is removed.
A queue has two different pointers: first and rear. In the “Enqueue” operation, the element is added in the index rear, then index rear is increasing until the maximum storage of the array. While in the “Dequeue” operation, an element is removed from index “front”, starting from the beginning and increasing until index from meets index rear.
The figure below illustrates the operations of stack and queue:



Summary:
Both stack and queue have different applications in various programs. Depends on the nature of the program, we can decide to use LIFO or FIFO to meet the requirements.
Appendix:

```
/*

============================================================================
 Name        : queuenp.c
 Instructor  : Taski Zourntos
 Modified by : Vy
 Version     :
 Copyright   : Your copyright notice
 Description : Hello World in C, Ansi-style
```

```
============================================================================
 */

#include <stdlib.h>
#include <stdbool.h>
#include <stdio.h>

#define L 1024
int q[L];
size_t rear = -1;
size_t front = -1;
void enque(int x)
{
            if(rear == L-1) // condition for overflow queue, which is when
index read = L-1
                  printf("Queue Overflow\n");
            else
            {
                  if(front == -1)
                  front = 0; //assign "pointer" front to index 0
                  rear = rear + 1; //increase the index of rear
                  q[rear] = x;//assign value to the index at q[rear]
                  printf("Index rear at:%d, Element enqueue is:%d\n", rear,
q[rear]);
            }

            return;
}
bool stackEmpty(void)
{
            bool result;
            if (front == -1 || front > rear)
            {
                  printf("Queue underflow.\n");
                  result = true;
            }
            else
                  result = false;
            return result;
}
void dequeue()
{
            printf("Front is at index: %d, Removed element is %d\n", front,
q[front]); /*after we run queue,
            front is at index 0*/
            front = front + 1; /*increment of front, front keep increasing
until meeting the condition in (stackEmpty)
            which is front == -1 or front > rear*/
}
int main(void) {
            int loadarr[] ={ 52, -29, 36, -821, 790, -650, 1125, 72, 0, 68,
33, 59 };
            size_t N = sizeof(loadarr) / sizeof(int); // size of the array,
size = total data size / size of an integer

            /* print out contents of array */
            printf("data to be loaded on to the stack:\n");
            for (size_t i = 0; i != N; ++i)
            {
                  printf("%d ", loadarr[i]); //in for loop from 0 to N-1,
showing the data from the stack.
            }
            printf("\n");

            // load queue

            printf("Enqueue elements: \n");
            for (size_t i = 0; i!=N; ++i)
            {
                  enque(loadarr[i]); //loading the element i into the queue.
            }

            //dequeue

            printf("Dequeue elements: \n");
            while(stackEmpty()==false)
            {
                  dequeue(); //after running enqueue, the front index is at
0.
            }

            // Exit normally*/
            return EXIT_SUCCESS;
}
```