#include <stdio.h>

#include <stdlib.h>

#define QUEUE\_EMPTY\_MAGIC 0xdeadbeef

typedef struct \_queue\_t {

int \*arr;

int rear, front, count, max;

} queue\_t;

queue\_t \*queue\_allocate(int n);

void queue\_insert(queue\_t \* q, int v);

int queue\_remove(queue\_t \* q);

int queue\_count(queue\_t \* q);

int queue\_is\_empty(queue\_t \* q);

void stack\_push(queue\_t \* q, int v) {

queue\_insert(q, v);

}

int stack\_pop(queue\_t \* q) {

int i, n = queue\_count(q);

int removed\_element;

for (i = 0; i < (n - 1); i++) {

removed\_element = queue\_remove(q);

queue\_insert(q, removed\_element);

}

removed\_element = queue\_remove(q);

return removed\_element;

}

int stack\_is\_empty(queue\_t \* q) {

return queue\_is\_empty(q);

}

int stack\_count(queue\_t \* q) {

return queue\_count(q);

}

int queue\_count(queue\_t \* q) {

return q->count;

}

queue\_t \*

queue\_allocate(int n) {

queue\_t \*queue;

queue = malloc(sizeof(queue\_t));

if (queue == NULL)

return NULL;

queue->max = n;

queue->arr = malloc(sizeof(int) \* n);

queue->rear = n - 1;

queue->front = n - 1;

return queue;

}

void queue\_insert(queue\_t \* q, int v) {

if (q->count == q->max)

return;

q->rear = (q->rear + 1) % q->max;

q->arr[q->rear] = v;

q->count++;

}

int queue\_remove(queue\_t \* q) {

int retval;

if (q->count == 0)

return QUEUE\_EMPTY\_MAGIC;

q->front = (q->front + 1) % q->max;

retval = q->arr[q->front];

q->count--;

return retval;

}

int queue\_is\_empty(queue\_t \* q) {

return (q->count == 0);

}

void queue\_display(queue\_t \* q) {

int i = (q->front + 1) % q->max, elements = queue\_count(q);

while (elements--) {

printf("[%d], ", q->arr[i]);

i = (i >= q->max) ? 0 : (i + 1);

}

}

#define MAX 128

int main(void) {

queue\_t \*q;

int x, select;

q = queue\_allocate(MAX);

do {

printf("\n[1] Push\n[2] Pop\n[0] Exit");

printf("\nChoice: ");

scanf(" %d", &select);

switch (select) {

case 1:

printf("\nEnter value to Push:");

scanf(" %d", &x);

stack\_push(q, x); /\* Pushing \*/

printf("\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\nCurrent Queue:\n");

queue\_display(q);

printf("\nPushed Value: %d", x);

printf("\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");

break;

case 2:

x = stack\_pop(q); /\* Popping \*/

printf("\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\nCurrent Queue:\n");

queue\_display(q);

if (x == QUEUE\_EMPTY\_MAGIC)

printf("\nNo values removed");

else

printf("\nPopped Value: %d", x);

printf("\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");

break;

case 0:

printf("\nQutting.\n");

return 0;

default:

printf("\nQutting.\n");

return 0;

}

} while (1);

return 0;

}