// Fig. 12.13: fig12\_13.c

// Operating and maintaining a queue

#include <stdio.h>

#include <stdlib.h>

// self-referential structure

struct queueNode {

char data; // define data as a char

struct queueNode \*nextPtr; // queueNode pointer

};

typedef struct queueNode QueueNode;

// function prototypes

void printQueue(QueueNode\* currentPtr);

//int isEmpty(QueueNode\* headPtr);

char dequeue(QueueNode\* \*headPtr, QueueNode\* \*tailPtr);

void enqueue(QueueNode\* \*headPtr, QueueNode\* \*tailPtr, char value);

void instructions(void);

// function main begins program execution

int main(void)

{

QueueNode\* headPtr = NULL; // initialize headPtr

QueueNode\* tailPtr = NULL; // initialize tailPtr

char item; // char input by user

instructions(); // display the menu

printf("%s", "? ");

unsigned int choice; // user's menu choice

scanf("%u", &choice);

// while user does not enter 3

while (choice != 3) {

switch(choice) {

// enqueue value

case 1:

printf("%s", "Enter a character: ");

scanf("\n%c", &item);

enqueue(headPtr, &tailPtr, item);

printQueue(headPtr);

break;

// dequeue value

case 2:

// if queue is not empty

if (headPtr != NULL) {

item = dequeue(&headPtr, &tailPtr);

printf("%c has been dequeued.\n", item);

}

printQueue(headPtr);

break;

default:

puts("Invalid choice.\n");

instructions();

break;

} // end switch

printf("%s", "? ");

scanf("%u", &choice);

}

puts("End of run.");

}

// display program instructions to user

void instructions(void)

{

printf ("Enter your choice:\n"

" 1 to add an item to the queue\n"

" 2 to remove an item from the queue\n"

" 3 to end\n");

}

// insert a node at queue tail

void enqueue(QueueNode\* \*headPtr, QueueNode\* \*tailPtr, char value)

{

QueueNode\* newPtr;

newPtr= malloc(sizeof(QueueNode));

if (newPtr != NULL) { // is space available

newPtr->data = value;

newPtr->nextPtr = NULL;

// if empty, insert node at head

if (\*headPtr == NULL) {

\*headPtr = newPtr;

}

else {

(\*tailPtr)->nextPtr = newPtr;

}

x

}

else {

printf("%c not inserted. No memory available.\n", value);

}

}

// remove node from queue head

char dequeue(QueueNode\* \*headPtr, QueueNode\* \*tailPtr)

{

char value = (\*headPtr)->data;

/\*

The error prevention code (headPtr != NULL)

is in function main however

you can also put it here

\*/

QueueNode\* tempPtr = \*headPtr;

\*headPtr = (\*headPtr)->nextPtr;

// if queue is empty

if (\*headPtr == NULL) {

\*tailPtr = NULL;

}

free(tempPtr);

return value;

}

// print the queue

void printQueue(QueueNode\* currentPtr)

{

// if queue is empty

if (currentPtr == NULL) {

puts("Queue is empty.\n");

}

else {

puts("The queue is:");

// while not end of queue

while (currentPtr != NULL) {

printf("%c --> ", currentPtr->data);

currentPtr = currentPtr->nextPtr;

}

puts("NULL\n");

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* (C) Copyright 1992-2015 by Deitel & Associates, Inc. and \*

\* Pearson Education, Inc. All Rights Reserved. \*

\* \*

\* DISCLAIMER: The authors and publisher of this book have used their \*

\* best efforts in preparing the book. These efforts include the \*

\* development, research, and testing of the theories and programs \*

\* to determine their effectiveness. The authors and publisher make \*

\* no warranty of any kind, expressed or implied, with regard to these \*

\* programs or to the documentation contained in these books. The authors \*

\* and publisher shall not be liable in any event for incidental or \*

\* consequential damages in connection with, or arising out of, the \*

\* furnishing, performance, or use of these programs. \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/