תרגיל בית 7 – מצביעים למצביעים

/$$$$$$$$ /$$ /$$ /$$$$$$$$ /$$ /$$   
|\_\_ $$\_\_/ |\_\_/| $$ |\_\_\_\_\_ $$/ | $$ /$$/   
 | $$ /$$$$$$ /$$$$$$ /$$$$$$ /$$| $$ /$$/ \ $$ /$$//$$$$$$   
 | $$ |\_\_\_\_ $$ /$$\_\_ $$ /$$\_\_ $$| $$| $$ /$$/ \ $$$$//$$\_\_ $$  
 | $$ /$$$$$$$| $$ \\_\_/| $$ \ $$| $$| $$ /$$/ \ $$/| $$ \ $$  
 | $$ /$$\_\_ $$| $$ | $$ | $$| $$| $$ /$$/ | $$ | $$ | $$  
 | $$| $$$$$$$| $$ | $$$$$$$| $$| $$ /$$/ | $$ | $$$$$$/  
 |\_\_/ \\_\_\_\_\_\_\_/|\_\_/ \\_\_\_\_ $$|\_\_/|\_\_/ |\_\_/ |\_\_/ \\_\_\_\_\_\_/   
 /$$ \ $$   
 | $$$$$$/   
 \\_\_\_\_\_\_/

## קוד:

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* INCLUDES & DEFINES \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <stdlib.h>

#include <stdio.h>

#define ERROR( x ) {fprintf(stderr,"%s\n",x); exit(EXIT\_FAILURE);}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* AUXILARY FUNCTION DECLERAION \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*

Compares 2 arrays formed using buildNumber.

@param two integer pointers to license plate array.

@return -1 if plate car1 < plate car2.

0 if plates are equal.

1 if plate car2 < plate car1.

\*/

int compLicense(int\* car1, int\* car2);

/\*\*

Reallocates new array of int\* with (new size = old size +1)

and creates a "gap" before input index i.

@param carArray = array of pointers to int.

size = array length.

i = index before which to create an empty element.

@return Array after adding gap at index i.

\*/

int\*\* incSort(int\*\* carArray,int size, int i);

void menu();

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* MAIN FUNCTION DECLERATIONS \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*

creates an array of individual digits from Integer.

@param users input Integer from within function.

@return int\* array of size USER\_INPUT\_LENGTH + 1.

- first cell is integer digit count.

- rest of cells are individual digits.

\*/

int\* buildNumber();

/\*\*

Adds a new plate from user input to Array.

@param CarArray = array of int\* pointing to plate numbers.

numOfCars = pointer to array length.

@return pointer to updated array with user plate added if successful,

unchanged pointer otherwise.

\*/

int\*\* addCar(int\*\* CarArray, int\* numOfCars);

/\*\*

Iteratively prints all plate numbers in plate array.

@param CarArray = array of int\* pointing to plate numbers.

numOfCars = array length.

@return Outputs to STDOUT

\*/

void printCarNumbers(int\*\* carArray, int numOfCars);

/\*\*

@param

@return

\*/

int\*\* deleteCar(int\*\* carArray, int\* carNumber);

/\*\*

Frees all dynamically allocated memory of plate array.

@param carArray = array of int\* pointing to plate numbers.

numOfCars = array length

@return NULL ptr;

\*/

int\*\* removeCarList(int\*\* carArray, int numOfCars);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* MAIN \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int main() {

menu();

return EXIT\_SUCCESS;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* MAIN FUNCTION IMPLEMENTATIONS \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int\* buildNumber() {

//initializers.

int plate = 0,temp =0, len = 0,\*arr = NULL;

printf("Enter vehicle number: ");

scanf("%d", &plate);

temp = plate;

//division by 10 loop to find integer digit count.

while (temp) {

temp /= 10;

++len;

}

//dynamic license plate array allocation and null check.

arr = (int\*)calloc(len +1,sizeof(int));

if (!arr) ERROR("Mem Error");

//insert remainders into license plate array.

for (int i = len; plate > 0; --i) {

arr[i] = plate % 10;

plate /= 10;

}

arr[0] = len;//put plate length in array first position.

return arr;

}

int\*\* addCar(int\*\* CarArray,int\* numOfCars) {

int\* tempPlate = buildNumber(), toAdd = 0;

// case car array is empty.

if (\*numOfCars == 0) {

CarArray = incSort(CarArray,\*numOfCars,0);

CarArray[0] = tempPlate;

++(\*numOfCars);

printf("Succesfully added car.\n");

return CarArray;

}

// case numOfCars >= 1

//check first and last

if (compLicense(tempPlate,CarArray[0]) == -1) {

CarArray = incSort(CarArray,\*numOfCars,0);

CarArray[0] = tempPlate;

++(\*numOfCars);

printf("Succesfully added car.\n");

return CarArray;

}

if (compLicense(CarArray[\*numOfCars - 1],tempPlate) == -1) {

CarArray = incSort(CarArray,\*numOfCars,\*numOfCars + 1);

CarArray[\*numOfCars] = tempPlate;

++(\*numOfCars);

printf("Succesfully added car.\n");

return CarArray;

}

//check rest of car array.

while (toAdd < \*numOfCars) {

if (compLicense(CarArray[toAdd],tempPlate) == -1 &&

compLicense(tempPlate,CarArray[toAdd+1]) == -1) {

CarArray = incSort(CarArray,\*numOfCars,toAdd + 1);

CarArray[toAdd + 1] = tempPlate;

++(\*numOfCars);

printf("Succesfully added car.\n");

return CarArray;

}

++toAdd;

}

// case car already exists.

printf("Car already exist in database\n");

return CarArray;

}

int\*\* deleteCar(int\*\* carArray, int\* numOfCars) {

//case array is empty.

if (\*numOfCars == 0) {printf("Car list is Empty.\n"); return NULL;}

//input user plate to remove.

int\* remove = buildNumber();

// search for plate in sorted array.

for (int i = 0; i < \*numOfCars; ++i) {

if (compLicense(remove,carArray[i]) == 0) {//if found.

free(carArray[i]);

//shift all cars after removed cars to left by 1.

for (int j = i; j < \*numOfCars - 1; ++j) {

carArray[j] = carArray[j + 1];

}// reallocate array to old size -1.

carArray = (int\*\*)realloc(carArray, (\*numOfCars - 1)\*sizeof(int\*));

--(\*numOfCars);

if (\*numOfCars != 0 && !carArray) ERROR("Mem Error");

free(remove);

printf("Successfully removed car.\n");

return carArray;

}

}

free(remove);

printf("\nCar not found... database unchanged\n");

return carArray;

}

void printCarNumbers(int\*\* carArray, int numOfCars) {

if (!numOfCars) {printf("Car list is empty.\n"); return;}

printf("\nCAR LIST:\n\*\*\*\*\*\*\*\*\n");

for (int j = 0; j < numOfCars; ++j) {

for (int i = 1; i <= carArray[j][0]; ++i) {

printf("%d",carArray[j][i]);

}

printf("\n");

}

printf("\n");

return;

}

int\*\* removeCarList(int\*\* carArray, int numOfCars) {

for (int i = 0; i < numOfCars; ++i) {

free(carArray[i]);

}

free(carArray);

return NULL;

}

/\*\*\*\*\*\*\*\*\*\*\* AUXILARY FUNCTION IMPLEMENTATIONS \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int compLicense(int\* car1, int\* car2) {

if (car1[0] < car2[0]) return -1;

if (car1[0] > car2[0]) return 1;

int len = car1[0];

for (int i = 1; i <= len; ++i) {

if (car1[i] < car2[i]) return -1;

if (car1[i] > car2[i])return 1;

}

return 0;

}

int\*\* incSort(int\*\* carArray,int size, int i) {

carArray = (int\*\*)realloc(carArray, sizeof(int\*)\*(size + 1));

if (!carArray) ERROR("Mem Error");

for (int j = size; j > i; --j) carArray[j] = carArray[j-1];

return carArray;

}

void menu() {

int choice = 0, \*\*carArray = NULL, size = 0;

printf("Menu:\n\

\*\*\*\*\*\*\n\

1 - Add Car.\n\

2 - Delete Car.\n\

3 - Print Car List.\n\

4 - Exit.\n");

while (choice != 4)

{

printf("Enter next menu choice: ");

scanf("%d", &choice);

switch (choice)

{

case 1:

carArray = addCar(carArray,&size);

break;

case 2:

carArray = deleteCar(carArray,&size);

break;

case 3:

printCarNumbers(carArray,size);

break;

default: break;

}

}

removeCarList(carArray,size);

return;

}

## פלט:

