מגישים:

אלמוג פדידה. ת.ז 315544437

מורן ארזי. ת.ז 200244945

חלק א' תיאוריה :

תרגיל 1-

מימוש תור בעזרת שתי מחסניות :

נתונות בידך 2 מחסניות בלבד : מחסנית A ומחסנית B ועליך לממש באמצעותן תור.

כלומר עליך להסביר כיצד יש להשתמש ב2 המחסניות על מנת לממש את 3 הפעולות המוגדרות על תור : הוצאה , הכנסה ובדיקה האם התור ריק?.

יש לפרט כיצד תתבצע כל פעולה לפי מדיניות התור **ע"י 2 המחסניות[באופן מילולי- רעיוני]**

פתרון:

הכנסה

הכנסת כל האיברים למחסנית א'
S1.push(x) – enq(x)

הוצאה

ריקון מחסנית א' לתוך מחסנית ב' (ולכן מתקבלת מחסנית א' הפוכה).
כעת נוציא את האיבר בראש מחסנית ב' (עם פופ) , ונרוקן את מחסנית ב' לתוך א' (עם push ו pop).

בדיקה האם התור ריק

נבדוק האם מחסנית א' ריקה שהיא המחסנית המכילה את המשתנים שנמצאים בתור.

<u>תרגיל 2</u> :
מה עושה האלגוריתם הבא?

```
Mistery(Queue q1, Queue q2){
        if(q1.empty() AND q2.empty())
                return true;
        if(q1.empty() OR q2.empty())
                return false;
        if(q1.dequeue ≠ q2.dequeue)
                return false;
        return mistery(q1, q2);
}
```

א. הדגם זוג אחד של תורים לא ריקים שעליהם יופעל האלגוריתם תעלומה, ויחזיר 'אמת' . הסבר.
ב. הדגם שני זוגות תורים לא ריקים עליהם יופעל האלגוריתם תעלומה ויחזיר 'שקר' מסיבות שונות. הסבר כל דוגמא.

פתרון:
א.

תור א' = (2,4,6,8)

תור ב' = (2,4,6,8)

הפונקציה תעלומה תחזיר אמת במקרה זה מכיוון שהאלגוריתם בודק הם זהים באורך ובתוכן ורק אז מחזיר אמת.

ב.

דוגמה 1:

תור א' = (2,4,6)

תור ב' = (2,4)

כאשר יוחזר שקר במקרה משום שהגדלים שלהם אינם שווים, כלומר אורך התור (יפול באיף השני כשיש OR).

דוגמה 2:

תור א' = (3,4,6)

תור ב' = (2,4,6)

כאן יוחזר שקר משום שאחד האיברים אינם זהים (נופל באיף השלישי).

חלק ב' מעשי :

דוגמת ריצה :

```
C:\Users\Almog Fadida\source\repos\assignment7_mivney\Debug\assignment7_mivney.exe
Almog Fadida ID 315544437, Moran Arzi ID 200244945


1. Add a patient to the queue
2. Remove a patient from queue
3. Remove a child from queue
4. Print the queues' ID numbers
5. Exit the program

Enter choice:
1
Please enter your ID number
100

1. Add a patient to the queue
2. Remove a patient from queue
3. Remove a child from queue
4. Print the queues' ID numbers
5. Exit the program

Enter choice:
1
Please enter your ID number
200

1. Add a patient to the queue
2. Remove a patient from queue
3. Remove a child from queue
4. Print the queues' ID numbers
5. Exit the program

Enter choice:
1
Please enter your ID number
300

1. Add a patient to the queue
2. Remove a patient from queue
3. Remove a child from queue
4. Print the queues' ID numbers
5. Exit the program
```

```
C:\Users\Almog Fadida\source\repos\assignment7_mivney\Debug\assignment7_mivney.exe
Please enter your ID number
1001

1. Add a patient to the queue
2. Remove a patient from queue
3. Remove a child from queue
4. Print the queues' ID numbers
5. Exit the program

Enter choice:
1
Please enter your ID number
500

1. Add a patient to the queue
2. Remove a patient from queue
3. Remove a child from queue
4. Print the queues' ID numbers
5. Exit the program

Enter choice:
1
Please enter your ID number
1002

1. Add a patient to the queue
2. Remove a patient from queue
3. Remove a child from queue
4. Print the queues' ID numbers
5. Exit the program

Enter choice:
4
The patient: 100 is in the queue
The patient: 200 is in the queue
The patient: 300 is in the queue
The patient: 1001 is in the queue
The patient: 500 is in the queue
The patient: 1002 is in the queue

1. Add a patient to the queue
```

```
C:\Users\Almog Fadida\source\repos\assignment7_mivney\Debug\assignment7_mivney.exe
5. Exit the program

Enter choice:
3
The child ID is : 1001

1. Add a patient to the queue
2. Remove a patient from queue
3. Remove a child from queue
4. Print the queues' ID numbers
5. Exit the program

Enter choice:
4
The patient: 100 is in the queue
The patient: 200 is in the queue
The patient: 300 is in the queue
The patient: 500 is in the queue
The patient: 1002 is in the queue

1. Add a patient to the queue
2. Remove a patient from queue
3. Remove a child from queue
4. Print the queues' ID numbers
5. Exit the program

Enter choice:
2
The patient has been removed from the queue.
His details were:
100

1. Add a patient to the queue
2. Remove a patient from queue
3. Remove a child from queue
4. Print the queues' ID numbers
5. Exit the program

Enter choice:
4
The patient: 200 is in the queue
```

```
Microsoft Visual Studio Debug Console
1. Add a patient to the queue
2. Remove a patient from queue
3. Remove a child from queue
4. Print the queues' ID numbers
5. Exit the program

Enter choice:
2
The patient has been removed from the queue.
His details were:
1002

1. Add a patient to the queue
2. Remove a patient from queue
3. Remove a child from queue
4. Print the queues' ID numbers
5. Exit the program

Enter choice:
2
ERROR: There are no patients in the queue so we can't remove from queue.
You must enter a patient into the queue and only then remove him from queue.

1. Add a patient to the queue
2. Remove a patient from queue
3. Remove a child from queue
4. Print the queues' ID numbers
5. Exit the program

Enter choice:
5

C:\Users\Almog Fadida\source\repos\assignment7_mivney\Debug\assignment7_mivney.exe (process 1116) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

(בחרנו במערך למימוש התור ובנוסף עשינו משתנה קבוע SIZE שמגדיר את גודל התור 20)

בחרנו להציג את התוכנית (main) בתוך סוויצ-קאיס בתוך דו וואיל , כאשר כל פעם אנו מחכים לקלט מהמשתמש כאשר :

1.הכנסת מטופל לתור

2.הוצאת מטופל מהרשימה והדפסת פרטיו

3.הוצאת הילד הראשון מהתור והדפסתו אם קיים

4.הדפסת כל התור ללא הוצאה

5. יציאה מהתפריט

בקובץ ה header קיימים ארבע תכונות שנשתמש בהם לאורך התוכנית והם:

1.head: שמציע על ראש התור

2. end: שמצביע על סוף תור

3.arrrq: המערך שנממש באמצעותו את התור

4.size: כמות המטופלים בתור כרגע

בקובץ cpp מימשנו את הפונקציות הבאות:

חלק א:

בנאי : שלא מקבל כלום ומאתחל את התכונות ל0 ואת המערך ל NULL בעזרת initQ

בנאי העתקה : העברת ואיתחול כל המשתנים לתור חדש

מחסל : מוחק את המערך ומשחרר זיכרון

חלק ב:

חמש פונקציות לשם מימוש התור:

אתחול הטור : אתחלנו את המערך לגודל 20 ואת משתניו ל NULL ואת שאר התכונות איתחלנו ל0 בכדי שיתחילו את המערך מתחילתו

הכנסה : בדקנו שהקלט מהמשתמש תקין וכן הודעה מתאימה (קלט > 0) ובדקנו שאם התור מלא תשלח הודעה מתאימה למשתמש שהתור מלא.

כאשר הקלט תקין הפונקציה תכניס את תעודת הזהות למערך במקום הend ותקדם את end ב1 ואת size ב1 בשביל לקדם את סוף הטור ואת כמות המטופלים בשביל התנאים.

(לקידום הטור השתמשנו במודולו גודל המערך בשביל תור מעגלי)

הוצאה : איתחלנו משתנה זמני temp שישמור לנו את תעודת הזהות שאנו עומדים להוציא

ואז בדקנו שהתור לא ריק , הכנסנו לtemp את ערך ראש התור ואיתחלנו את ראש התור לNULL

(בשביל למחוק את המטופל בראש התור) וקידמנו את head ב1 (עם מודולו)

הקטנו את כמות המטופלים (size) והחזרנו את temp

במקרה שהתור ריק החזרנו NULL

האם הטור ריק ?: בדקנו האם המטופל בראש התור שווה לNULL והחזרנו אמת כי זה מעיד על כך שהתור ריק החזרנו החזרנו שקר.

הצצה : בדקנו האם התור לא ריק והחזרנו את ערך ראש התור ללא הוצאה

אחרת החזרנו NULL כי אין מטופל בראש התור.

חלק ג:

פונקציות נוספות

הסרת מטופל מהתור : בדיקה האם התור לא ריק

אם כן מדפיס הודעה מתאימה

אם לא , מדפיס את פרטי המטופל בראש הטור ומוחק אותו מהטור

הדפסת הטור : השתמשנו בתור עזר בשביל לא לשנות את התור המקורי בעת ההדפסה

הפונקציה בלולאה תבדוק שהמערך המקורי לא ריק ותשתמש בהצצה בשביל להכניס לתור

העזר ללא פגיעה במקורי ואז תדפיס את פרטי המטופל בראש התור המקורי ותוציא אותו

לאחר שעברה על כל התור המקורי והדפיסה את כל המטופלים

היא תעבור בלולאה שתבדוק שהמערך עזר לא ריק ותוציא מטור העזר ותכניס לתור המקורי

בכך תשמור על הסדר והנתונים של התור המקורי.

הוצאת ילד: בעזרת משתנה עזר ותור עזר

הפונקציה בלולאה תבדוק שהמערך המקורי לא ריק וגם שתעודת הזהות קטנה מ999 (בעזרת

הצצה בשביל להגיע למטופל שהוא ילד) ותכניס את כל המבוגרים לתור העזר ותוציא מהטור המקורי

במקרה שנמצא ילד נשתמש במשתנה העזר בשביל לשמור אותו ונוציא אותו מהטור

(במקרה שלא נמצא ילד המשתנה ישמור 0 וכל המבוגרים יעברו למערך העזר)

ולאחר מכן נמשיך להעביר מטופלים (ילדים ומבוגרים) לטור העזר

אחר כך נעבור בלולאה שתבדוק שמערך העזר לא ריק בשביל להכניס את כל המטופלים שהיו לתור

המקורי לפי הסדר אבל ללא הילד הראשון ומחזירה את הילד שמצאנו (או 0 אם לא מצאנו)

הקוד :

Header:

```
//Almog Fadida ID 315544437, Moran Arzi ID 200244945

#pragma once
#include <iostream>
using namespace std;
#define SIZE 20   //size of the queue

class yeladudes {
```

```
private:

	int head; //point to the head of the queue
	int end;  //point to the end of the queue
	int* arrq;  //the arr that we use as a queue
	int size; //The number of elements that we obtain currently inside the queue

public:

	//constractor, copy constractor and destractor

	yeladudes();
	yeladudes(const yeladudes& other);
	~yeladudes();



	//implements queue functions with an array

	void initQ();
	void enqueue(const int ID_patient);
	int dequeue();
	bool empty();
	int front();

	//Functions especially for this program

	void removePatientFromQueue();
	void printQ();
	int popchild();

};
```

Cpp:

```
//Almog Fadida ID 315544437, Moran Arzi ID 200244945

#include "yeladudes.h"

//constractor, copy constractor and destractor

yeladudes::yeladudes() {
	initQ();

}//end constractor yeladudes()


yeladudes::yeladudes(const yeladudes& other){

	delete[] arrq;
	arrq = other.arrq;
	head = other.head;
	end = other.end;
	size = other.size;

}//end copy constractor yeladudes()


yeladudes::~yeladudes() {
	delete[] arrq;

}//end destractor ~yeladudes()
```

```
//implements queue functions with an array

void yeladudes::initQ() {

	arrq = new int[SIZE] {NULL};  //initilazing the arr (as queue) with size 20 and
each one will be NULL at the beginning

	head = 0; //In the beginning head and end are at the same place, the start.

	end = 0;

	size = 0; //no elements are inside at first

}


void yeladudes::enqueue(const int ID_patient) {

	if (ID_patient <= 0)
		cout << "not a valid ID" << endl; //ID must be bigger than 0.

	else {
		if (size == SIZE) { //The queue is full and there's no place to enqueue
			cout << "no more room in queue" << endl;
		}

		else { //if (end != head) - there's room in queue
			arrq[end] = ID_patient;  //we put ID at the value of the end

			end = (end + 1) % SIZE; //and than change end place one step more
in the queue (mod because it's circular queue,
			//means if we're at the 20 box (the end) so it goes to the
beginning again after that.

			size++; //increase size because we enqueue one
		}
	}
} //end enqueue(const int ID_patient)


int yeladudes::dequeue() { //dequeue will remove the first element in queue and
increase the head

	int temp;

	if (!(this->empty())) { //if queue isn't empty
		temp = arrq[head]; //put the value of the first element in queue inside
temp
		arrq[head] = NULL; //and delete the value now

		head = (head + 1) % SIZE; //increase head like before (in mod for a
circular manner)

		size--; //decrease size

		return temp; //return the value that was in the head
	}

	return NULL; //if queue is empty so return null and don't remove anything.
```

```
} //end dequeue()


bool yeladudes::empty() {

	if (arrq[head] == NULL) { //if the first element is null it means the queue is
empty
		return true;
	}
	else {
		return false;
	}

} //end empty()

int yeladudes::front() {

	if (!(this->empty())) { //if queue isn't empty pick at the head (value).

		return arrq[head];
	}

	return NULL; //if queue is empty show Null because this is the element head

} //end front()



//Functions especially for this program

void yeladudes::removePatientFromQueue() {
	if (!this->empty()) { // Check if a parient in the queue
		cout << "The patient has been removed from the queue." << endl;
		cout << "His details were: " << endl;
		cout << this->dequeue() << endl;  // We print the patient details and
remove him from the queue

	}
	else { //If the queue is empty
		cout << "ERROR: There are no patients in the queue so we can't remove
from queue." << endl;
		cout << "You must enter a patient into the queue and only then remove
him from queue."<< endl;
	}

} //end removePatientFromQueue function



int yeladudes::popchild() {

	int firstChild;
	yeladudes* temp = new yeladudes(); // Make a new queue

	while (!(this->empty()) && (this->front()) < 1000 ) { //Until we encounter a
patient with ID bigger then 999 (child)
		temp->enqueue(this->dequeue());                                //we take
the head of the main queue (adult) and put him in a temp queue
	}
		//and we remove him from the main queue.
```

```
	if (!this->empty()) //We found a child
		firstChild = this->dequeue();
	else                          //We did not find a child
		firstChild = 0;

	while (!this->empty()) { // We countinue to remove from main queue and add it
to the temp queue
		temp->enqueue(this->dequeue()); //so that all the main queue is in the
temp without the child (we save the main queue order).
	}


	while (!temp->empty()) { //We return the main queue to its original order
without the child.
		this->enqueue(temp->dequeue());
	}

	delete temp;

	return firstChild;


} //end popchild function



void yeladudes::printQ() {

	yeladudes* temp = new yeladudes(); //we use a second arr as queue in order to
implements print function without damaging the main
                                                                        //queue order.

	while(!this->empty()){
		temp->enqueue(this->front()); //we add the head of the main queue
without removing it and we add it to the temp queue
		cout << "The patient: " << this->dequeue() << " is in the queue" <<
endl; //we print the patient that in the queue.
	}

	while (!temp->empty()) { //We return the main queue to its original order.
		this->enqueue(temp->dequeue());
	}


                                                                    } //end printQ function
```

:main

```
//Almog Fadida ID 315544437, Moran Arzi ID 200244945


#include "yeladudes.h"

//We chose switch case inside do while in order to appley this program.
//The user choice will get in showMenuGetNumChoice function after showing him the menu
there (after each choice)
//until he'll press 5 (to exit the program). showMenuGetNumChoice function is located
after the main function.

int main() {
```

```
	cout << "Almog Fadida ID 315544437, Moran Arzi ID 200244945" << endl;

	cout << "" << endl;

	int showMenuGetNumChoice();

	yeladudes* queue = new yeladudes();

	int choice;

	do {
		choice = showMenuGetNumChoice();

		switch (choice) {

		case 1:

			int ID_patient;
			cout << "Please enter your ID number" << endl;
			cin >> ID_patient;
			queue->enqueue(ID_patient);
			break;

		case 2:

			queue->removePatientFromQueue();
			break;

		case 3:
			int child;
			child = queue->popchild(); //remove the first child in queue

			if (child == 0)
				cout << "no children in queue " << endl;

			else
				cout << "The child ID is : " << child << endl; //print
first child ID


			break;

		case 4:

			queue->printQ();
			break;

		}

	} while (choice != 5); //case 5 doesn't exist so it's the exit number from the
menu and program.

	delete queue;

	return 0;
}

int showMenuGetNumChoice() {

	int choice;
	//show menu
```

```
    cout << "" << endl;
    cout << "1. Add a patient to the queue" << endl;
    cout << "2. Remove a patient from queue" << endl;
    cout << "3. Remove a child from queue" << endl;
    cout << "4. Print the queues' ID numbers" << endl;
    cout << "5. Exit the program" << endl;

    cout << "" << endl;
    //get choice
    cout << "Enter choice: " << endl;
    cin >> choice;

    return choice;
}
```