**מבוא למדעי המחשב – 150005**

**תרגיל בית מספר 8**

**הקצאת זיכרון דינמית.**

1. השתמש בשמות משמעותיים עבור המשתנים.
2. יש לתעד את התכנית (כולל הערה לפני התכנית הראשית, המסבירה את מטרתה ואת פעולתה).
3. הקפד על קְרִיאוּת התכנית ועל עימוד (Indentation).
4. הקפד לבצע בדיוק את הנדרש בכל שאלה.
5. יש לבצע בדיקת תקינות קלט, כרגיל (בתרגיל זה, קלט לא תקין ייתכן במקרים הבאים: מספר קו, מספר תחנות, משך נסיעה (שלילי) וכן בחירת ערך שאינו בין 1 ל 7 בתפריט).
6. יש להגיש את התרגיל על פי ההוראות הרשומות במודל.

**הערה חשובה: לעיתים הבודק האוטומטי לא מקבל את פקודת delete – ולכן לא יתקבל ציון 80. במקרה זה, יש לסמן כהערה את פקודת ה delete לפני העלאה לבדיקה נוספת (אך לא למחוק אותה. פקודה חסרה לא תקבל ציון בבדיקה הידנית).**

**הערה חשובה נוספת: התרגיל בודק שימוש במצביעים. לכן חובה להשתמש במצביעים בפונקציות הנדרשות בתרגיל זה. פתרון אשר ישתמש באינדקסים של מערכים לא יקבל את מירב הנקודות.**

חברת האוטובוסים בס-לב, רוצה למחשב את המידע לגבי קווי האוטובוסים שלה.

לחברה ישנו מספר משתנה של קווי אוטובוס. עבור כל קו החברה מחזיקה במידע מהו מספר הקו, מספר התחנות במסלול הנסיעה של הקו, ומהו משך הנסיעה המשוער החל מתחנת היציאה ועד לתחנת ההורדה.

החברה מעוניינת באפשרות לקבל את המידע הבא, על סמך הנתונים שקיימים במערכת:

* בהינתן מספר קו – האם הקו קיים במערכת.
* מהו זמן הנסיעה הממוצע של האוטובוסים בחברה.
* מהו מספר התחנות הממוצע של קו אוטובוס באוטובוסי החברה.
* מהו הקו אשר משך הנסיעה שלו הוא הקצר ביותר.
* הדפסת כל הנתונים הקיימים במערכת.

בנוסף, החברה מעוניינת שהמידע במערכת יוכל להיות דינמי, ולבצע בעת הצורך את השינויים הבאים:

* הוספת קו למערכת
* הורדת קו מהמערכת

לצורך כך, כתוב תכנית, אשר כוללת את הנתונים והפעולות הבאים:

מסד הנתונים של החברה ישמר במערך דו ממדי בשם buses. מספר השורות במערך הוא (בדיוק) כמספר הקווים הקיימים. ומספר העמודות הוא 3. כאשר, עבור כל קו - העמודה הראשונה מכילה את מספר הקו, השנייה את מספר התחנות במסלול, והשלישית – את משך הנסיעה. (להזכירך: מערך דו ממדי, הוא למעשה מערך חד ממדי של מצביעים. כאשר - כל אחד מהמצביעים מצביע על מערך חד ממדי נוסף).  
המטריצה תישמר בכל עת כשהיא ממוינת לפי מספרי הקווים.

התכנית מכילה את הפונקציות הבאות:

1. **addLine -** הוספת קו אוטובוס חדש למערכת. הפונקציה מקבלת את המטריצה buses ואת גודלה(את מספר הקווים הקיימים כרגע במערכת), וכן – נתונים על הקו החדש: מספר הקו, מספר תחנות ומשך נסיעה. על הפונקציה להוסיף את המידע למערך buses. (שים לב: המשמעות היא, בין השאר, שאורך מערך המצביעים גדל ב-1. כמו כן – גם המערך buses החדש יהיה ממוין לפי מספרי הקווים לאחר הכנסת הקו החדש).
2. **deleteLine -** הורדת קו אוטובוס קיים מהמערכת. הפונקציה מקבלת את המטריצה buses ואת גודלה(את מספר הקווים הקיימים כרגע במערכת), וכן – את מספר הקו להורדה מהמערכת. על הפונקציה למחוק את כל המידע על קו זה מהמערך buses. (המשמעות היא, בין השאר, שאורך מערך המצביעים קטן ב- 1).   
   שים לב! במידה שהקו אותו ביקש המשתמש למחוק לא קיים במערכת, הפונקציה תשאיר את המערך **ללא שינוי** (**אין** הודעת שגיאה ו**אין** קליטת קלט חדש)
3. **print -** הדפסת כל הנתונים שקיימים במערכת. הפונקציה מקבלת את המטריצה buses ואת גודלה(את מספר הקווים הקיימים כרגע במערכת). על הפונקציה להדפיס את כל הנתונים במטריצה (כלומר: עבור כל קו יודפס: מספר הקו, מספר התחנות בקו ומשך זמן הנסיעה)

**בפונקציה זו יש להתשמש במצביעים בלבד!**! (כלומר, ללא שימוש באינדקסים של מערכים)

1. **search -** חיפוש קו אוטובוס במערכת. הפונקציה מקבלת את המטריצה buses ואת גודלה(את מספר הקווים הקיימים כרגע במערכת).וכן – את מספר הקו לחיפוש. על הפונקציה לבדוק האם מספר הקו קיים במערכת. אם הקו קיים, הפונקציה תחזיר את מספר השורה בה נמצאים נתוני הקו במערכת. במידה והקו לא קיים במערכת, יוחזר הערך -1.
2. **averageTravel -** חישוב זמן הנסיעה הממוצע (כלומר: סכום זמני הנסיעות של כל הקווים, מחולק במספר הקווים הקיימים). הפונקציה מקבלת את המטריצה buses ואת גודלה(את מספר הקווים הקיימים כרגע במערכת). על הפונקציה להחזיר את ממוצע זמן הנסיעה של מסלולי אוטובוסי החברה.

**בפונקציה זו יש להתשמש במצביעים בלבד!**! (כלומר, ללא שימוש באינדקסים של מערכים)

1. **averageStops -** חישוב מספר התחנות הממוצע (כלומר: סכום מספר התחנות של כל הקווים, מחולק במספר הקווים הקיימים). הפונקציה מקבלת את המטריצה buses ואת גודלה(את מספר הקווים הקיימים כרגע במערכת). על הפונקציה להחזיר את ממוצע מספר התחנות של אוטובוסי החברה, כשהוא מעוגל כלפי למטה. (אפשר לבצע את זה על ידי המרה ל-int).
2. **shortest –** מציאת הקו עם זמן הנסיעה המינימאלי. הפונקציה מקבלת את המטריצה buses ואת גודלה(את מספר הקווים הקיימים כרגע במערכת) על הפונקציה להחזיר את מספר הקו שמשך הנסיעה בו הוא הקצר ביותר. במידה והמערך ריק הפונקציה תחזיר -1.

**תזכורת: יש לשחרר את הזיכרון הדינמי עם סיום השימוש בו!**

בתוכנית הראשית יש להצהיר על המערך. לאחר מכן יש לכתוב פקודה לביצוע חוזר (לולאה) המאפשרת למשתמש לבחור את הפעולה הרצויה (ע"י הכנסת מספר בין 0 ל- 7). עבור כל בקשה מהמשתמש, התכנית הראשית תקרא לפונקציה המתאימה, כדלהלן. כאשר המשתמש יבחר בפעולה מספר 0, התכנית תסיים את פעולתה.

כאשר המשתמש בוחר בפעולה 1, התכנית הראשית תקלוט את פרטי הקו החדש, ותקרא לפונקציה addLine, ולאחר מכן לפונקציה print.

כאשר המשתמש בוחר בפעולה 2, התכנית הראשית תקלוט את מספר הקו למחיקה, תקרא לפונקציה deleteLine, ולאחר מכן לפונקציה print.

כאשר המשתמש בוחר בפעולה 3, התכנית הראשית תקרא לפונקציה print

כאשר המשתמש בוחר בפעולה 4, התכנית הראשית תקלוט את מספר הקו לחיפוש, ותקרא לפונקציה search , ותדפיס את הערך המוחזר ממנה.

כאשר המשתמש בוחר בפעולה 5, התכנית הראשית תקרא לפונקציה averageTravel, ותדפיס את הערך המוחזר ממנה.

כאשר המשתמש בוחר בפעולה 6, התכנית הראשית תקרא לפונקציה averageStops , ותדפיס את הערך המוחזר ממנה.

כאשר המשתמש בוחר בפעולה 7, התכנית הראשית תקרא לפונקציה shortest, ותדפיס את הערך המוחזר ממנה.

**הנחיות נוספות:**

* **עבור הפעולה 1** (הוספת קו)

התכנית תדפיס:

enter the line to add

enter the number of stops

enter the duration of the ride

לאחר כל שורת פלט, התכנית תקלוט ערך מתאים (סך הכל שלושה ערכים: מספר הקו להוספה, מספר התחנות שלו ומשך הנסיעה במסלול)

* **עבור הפעולה 2** (מחיקת קו)

התכנית תדפיס:

enter the line to delete

לאחר מכן התכנית תקלוט ערך מתאים (מספר הקו למחיקה)

**לאחר כל הוספה או מחיקה על התוכנית להדפיס את המערך המעודכן**

* **עבור הפעולה 3** (הדפסה)

יודפסו נתוני המערך בצורה הבאה:

* על התוכנית להדפיס את הנתונים (מספר קו, מספר תחנות, משך נסיעה) כך שהנתונים של כל קו מופיעים באותה שורה (מופרדים באמצעות רווחים), וכל קו מופיע בשורה נפרדת.

לדוגמא: עבור שני קווים 2 ו5 כך שבקו 2 היה 10 תחנות וזמן נסיעה של 55 דקות ובקו 2 היו 12 נסיעות עם זמן נסיעה של 70 דקות, הפלט יהיה:

2 10 55

5 12 70

* **עבור הפעולה 4** (חיפוש)

התכנית תדפיס:

enter the line to search for

לאחר מכן התכנית תקלוט ערך מתאים (מספר הקו המבוקש). במידה שהמשתמש הכניס קו שקיים במערך, התוכנית תדפיס את מספר השורה שבה נמצאים נתוני הקו במטריצה. במידה והקו אינו קיים במערכת, התכנית תדפיס -1  
(שים לב: הפונקציה מחזירה את הערך להדפסה והתוכנית הראשית מדפיסה אותו)

* **עבור הפעולה 5** (חישוב זמן ממוצע של נסיעות )

התוכנית תדפיס את הערך המוחזר מהפונקציה averageTravel.

(שים לב: הפונקציה מחזירה את הערך להדפסה ותוכנית הראשית מדפיסה אותו)

* **עבור הפעולה 6** (חישוב ממוצע מספר התחנות )

התוכנית תדפיס את הערך המוחזר מהפונקציה averageStops.

(שים לב: הפונקציה מחזירה את הערך להדפסה ותוכנית הראשית מדפיסה אותו)

* **עבור הפעולה 7** (שם הקו בעל משך הנסיעה המינימלי)

התוכנית תדפיס את הערך המוחזר מהפונקציה shortest.

(שים לב: הפונקציה מחזירה את הערך להדפסה ותוכנית הראשית מדפיסה אותו)

* **עבור הפעולה 0** (יציאה )

יציאה מהתוכנית, ללא כל הדפסה נוספת.

* **עבור כל מספר אחר (שגיאת קלט)**

התכנית תדפיס את ההודעה הבאה:

ERROR

enter 0-7 .

המלצה: תכנן את התוכנית – על ידי Top Down Design - ראשית כתוב את התוכנית הראשית, ולאחר מכן כתוב את הפונקציות. ניתן להשתמש בשלד התוכנית הראשית המתוארת למטה, עם שינויים כרצונך. כמובן, שחובה עליך לכתוב את הפונקציות להן קוראת התוכנית הראשית.

בנוסף לתוכנית הראשית נתון לכם פונקציית print.

עליכם להשלים רק את הפוקציות האחרות. חובה להשתמש אך ורק במצביעים בטיפול במערך בפונקציות שבם התבקשתם לכך!

הערות :

* לפני שתתחילו לכתוב תוכנית כדאי לעבור על התוכנית הראשית להבין בדיוק מה התוכנית אמורה לעשות.
* כדאי לעבור על הפונקציה print להבין בדיוק איך משתמשים במצביעים כדי לבצע את דרישות התרגיל.
* מומלץ להגיש גם תכנית חלקית. אם לא הצלחת להגיע לתרגיל מושלם מומלץ להגיש תרגיל חלקי -. תכנית כזו תקבל ציון מתאים בבדיקה הידנית.
* עבור הפונקציות addbuses ו-deletebuses  שים לב להעביר את המצביע כ-reference אחרת השינוי שנעשה בפונקציה לא יבוא לידי ביטוי בתכנית הראשית.  
  לדוגמה addLine(int \*\*& buses)

**בהצלחה!**

#include <iostream>

using namespace std;

enum choices {EXIT, ADD, DELETE, PRINT, SEARCH, AVG\_TRAVEL, AVG\_STOPS, SHORTEST\_TRAVEL};

void addLine(int\*\* & buses,int & numBuses,int line,int stops,int durationOfRide);

void deleteLine(int\*\* & buses, int & numBuses,int line);

int search(int\*\* buses, int numBuses, int line);

float averageTravel(int\*\* buses, int numBuses);

float averageStops(int\*\* buses, int numBuses);

int shortest(int\*\* buses, int numBuses);

// prints out the buses that are saved in the database

void print(int \*\* buses, int numBuses) {

for (int i = 0; i<numBuses; i++) { // for each bus that is present in the database

//int \* oneBus = \*(buses + i);// store the bus’ data in oneBus

for (int j = 0; j<3; j++) { // prints the 3 pieces of data of the bus

cout << \*(\*(buses + i) + j)<<" ";

//cout << \*(oneBus + j);

//cout << " ";

}

cout << endl;

}

}

int main(){

int \*\* buses;

int numBuses = 0;

int line;

int stops;

int durationOfRide;

int choice;

do {

cout << "enter 0-7" << endl;

cin >> choice;

switch(choice){

case ADD: // add a line to the array of buses

cout << "enter the line to add" << endl;

cin >> line;

cout << "enter the number of stops" << endl;

cin >> stops;

cout << "enter the duration of the ride" << endl;

cin >> durationOfRide;

if (numBuses == 0)

//buses = new int \*[1];

addLine(buses, numBuses, line, stops, durationOfRide);

print(buses,numBuses);

break;

case DELETE: // delete a line from the array of buses cout << "enter the line to delete" << endl;

cin >> line;

deleteLine(buses, numBuses, line);

print(buses,numBuses);

break;

case PRINT: // print all lines

print(buses,numBuses);

break;

case SEARCH : // search for a particular line

cout << "enter the line to search for" << endl;

cin >> line;

cout << search(buses, numBuses, line)<<endl;

break;

case AVG\_TRAVEL : // calculate average travel time of all buses

cout << averageTravel(buses, numBuses)<<endl;

break;

case AVG\_STOPS: // calcuate average stops of all buses

cout << averageStops(buses, numBuses)<<endl;

break;

case SHORTEST\_TRAVEL:// calculate the bus with the shortest travel time

cout << shortest(buses,numBuses)<<endl;

break;

case EXIT: break; // exit the program

default: cout << "ERROR" << endl; }// switch

} while (choice != 0);

for(int i=0; i < numBuses; i++)

delete[] buses[i];

if (numBuses > 0)

delete [] buses;

return 0;

}