**רשימות מקושרות**

**שאלה 1**

נתונים המבנים הבאים לייצוג רשימה מקושרת של תווים:

typedef struct list\_node

{

char\* dataPtr;

struct list\_node\* next;

}ListNode;

typedef struct list

{

ListNode\* head;

ListNode\* tail;

}List;

נתונה **רשימה מקושרת** שנתוניה הם ספרות, ואותיות אנגליות קטנות בלבד. הרשימה המקושרת במקורה הכילה את המידע הבא על תלמיד: שמו הפרטי של התלמיד (המופיע באותיות קטנות) אות אחר אות ברשימה, ואחריו מופיע ציונו הממוצע (הניתן בספרות המייצגות מספר חיובי שלם) סיפרה אחר סיפרה ברשימה. לרוע המזל, כתוצאה מבאג במחשב, נשזרו התווים אלו באלו, והתקבלה רשימה מעורבלת.

דוגמה: רשימה שבמקור הכילה את התווים muℓℓy94 כאשר m היה הנתון בצומת הראשון, u היה הנתון בצומת השני, וכ"ו (כלומר שמו הפרטי הוא muℓℓy, וציונו 94), שונתה וכעת היא מכילה את התווים: m9uℓ4ℓy (כאשר m הוא הנתון בצומת הראשון, 9 הוא הנתון בצומת השני, וכ"ו).

שימו לב כי סדר התווים והספרות המקורי נשמר בתוך הערבול. (לדוגמא, u יבוא אחרי m, ℓ יבוא אחרי u, וכו').

כתבו פונקציה המקבלת **רשימה** מעורבלת ומחזירה מבנה לייצוג תלמיד (שיוגדר להלן) המכיל את שמו הפרטי ואת ציונו הממוצע של התלמיד המתקבלים מהפרדת המילים ברשימה המעורבלת.

ה – prototype של הפונקציה הוא:

Student unScramble (List lst);

כאשר מבנה של תלמיד מוגדר להלן:

typedef struct student

{

List first;  
 int grade;

} Student;

* השדה first הוא רשימה המייצגת את שמו הפרטי של הסטודנט אות אחר אות.
* והשדה grade הוא מספר המייצג את ממוצע ציוניו של הסטודנט.

הערות:

אין להשתמש בהקצאה דינאמית, אלא לשנות מצביעים ברשימת הקלט ולשחרר את הצמתים שהכילו ספרות.

**שאלה 2**

נתונה ההגדרה הבאה לייצוג רשימה מקושרת של מספרים שלמים:

typedef struct listNode{

int\* dataPtr;

struct listNode\* next;

}ListNode;

typedef struct list

{

ListNode\* head;

ListNode\* tail;

}List;

כתבו את הפונקציה הבאה, בארבע גרסאות:

List merge(List lst1, List lst2);

הפונקציה מקבלת כקלט lst1 ו-lst2, שתי רשימות מקושרות **ממוינות** בסדר **יורד** של ערכים עליהם מצביעים השדות dataPtr.   
על הפונקציה למזג את שתי הרשימות לרשימה ממוינת אחת המכילה את כל האיברים של רשימות הקלט. על הפונקציה להחזיר את הרשימה הממוזגת.

א. בגרסא זו, הפונקציה איננה רקורסיבית ועליה ליצור **רשימה חדשה**.   
הפונקציה לא תשנה הצבעות ברשימות lst1 ו- lst2 (בסוף ריצת הפונקציה שתי הרשימות נותרות ללא שינוי).

ב. בגרסא זו, הפונקציה איננה רקורסיבית ועליה למזג את שתי הרשימות לרשימה אחת ממוינת ע"י **שינוי מצביעים** בלבד (ללא שימוש בהקצאות חדשות).

ג. בגרסא זו, הפונקציה רקורסיבית ועליה ליצור **רשימה חדשה**.

ד. בגרסא זו, הפונקציה רקורסיבית ועליה למזג את שתי הרשימות ע"י **שינוי מצביעים**.

הערה: בסעיפים ג' ו- ד' הפונקציה אינה חייבת להיות רקורסיבית בעצמה, היא יכולה להיות פונקציה העוטפת פונקציה רקורסיבית (פונקציה אשר מפעילה לפונקציה רקורסיבית).

**שאלה 3**

מעוניינים לשמור נתונים על נקודות במישור ששיעורי הקורדינטות שלהן הם מספרים שלמים. על-מנת לחסוך במקום הציעו לשמור את הנקודות במבנה הבא:

**5**

**3**

**8**

**7**

**4**

**9**

**-4**

**9**

**9**

***שיעור הקורדינטה Y***

***שיעור הקורדינטה X***

המבנה לעיל שומר את הנקודות:

(5,3), (5,8), (7,4), (9,-4), (9,9), (9,9)

שימו לב, שנקודה יכולה להופיע יותר מפעם אחת.

החיסכון בא לידי ביטוי בכך שהקורדינטה X נשמרת פעם אחת בלבד עבור אוסף נקודות אשר להן שיעור קורדינטה X זהה ואשר נבדלות רק בערכי הקורדינטה Y. שימו לב כי המבנה הנ"ל עושה שימוש בשני סוגים שונים של רשימות אשר כל אחת מהן **ממויינת**.

typedef struct XlistNode{

…

} XListNode;

typedef struct YlistNode{

…

} YListNode;

typedef struct Ylist

{

…

} YList;

typedef struct list

{

…

} List;

1. כתבו את הפונקציה

unsigned int getPairOccurrences(List coord\_list,

int x, int y);

אשר סופרת ומחזירה כמה פעמים מופיעה הנקודה (x,y) ב-.coord\_list

1. כתבו את הפונקציה

unsigned int getYOccurrences(List coord\_list, int y);

אשר סופרת ומחזירה כמה נקודות (\*,y) נמצאות ב-.coord\_list כלומר כאלו אשר שיעור הקורדינטה Y שלהן שווה לפרמטר y.

1. כתבו את הפונקציה

unsigned int getXOccurrences(List coord\_list, int x);

אשר סופרת ומחזירה כמה קורדינטות (x,\*) נמצאות ב-.coord\_list כלומר כאלו אשר שיעור הקורדינטה X שלהן שווה לפרמטר x.

1. כתבו את הפונקציה

int insertCoordinate(List \*coord\_list, int x, int y);

אשר מוסיפה ל- coord\_listאת הנקודה (x,y). אם הנקודה נמצאת כבר, הפונקציה תחזיר 1, אחרת הפונקציה תחזיר 0. בכל מקרה, הפונקציה תוסיף את הנקודה.

1. כתבו את הפונקציה

int removeCoordinate(List \*coord\_list, int x, int y);

אשר מוחקת מ- coord\_listאת הנקודה (x,y). אם הנקודה מופיעה יותר מפעם אחת ב- coord\_list, יש למחוק רק את אחד המופעים. במקרה וזו הנקודה היחידה עם שיעור ה-X שצויין, יש למחוק גם את התא המתאים ברשימת שיעורי קורדינטות ה-X.

לדוגמא, לאחר הפעלת

removeCoordinate( &coord\_list, 7, 4);

על המבנה שבעמוד הקודם, יתקבל המבנה

**5**

**3**

**8**

**9**

**-4**

**9**

**9**

*שיעור*

*הקורדינטה*

*Y*

*שיעור הקורדינטה X*

על הפונקציה removeCoordinate להחזיר את אחד מהערכים הבאים:

אם הנקודה לא נמצאה, הפונקציה תחזיר 1,

אם יש מופע נוסף לנקודה, הפונקציה תחזיר 2,

אם זו הייתה הקורדינטה היחידה עם השיעור X שצויין, הפונקציה תחזיר 3,

בכל מקרה אחר, הפונקציה תחזיר 0.