פרויקט מסכם תכנות מתקדם – תשע"ט סמסטר קיץ

שאלה 1

נתונה ההגדרה הבאה של תא רשימה מקושרת של מילים וכמות המופעים שלהן:

```
typedef struct dlistNode
{
     char *str;
     unsigned short frequency;
     struct dlistNode *next;
} DListNode;
```

כתבו את הפונקציה:

unsigned int InsertToArrayOfLists(char *word, char freq,
DListNode ***word_lists_arr)

הפונקציה מקבלת מחרוזת **word** שמכילה מילה אחת וכמות מופעים **freq** (אשר יכולה להיות גם שלילית). המילה מורכבת מהאותיות AB...Z והאותיות ab...z בלבד. כמו-כן, הפונקציה מקבלת *by ref* מערך **word_lists_arr** אשר כל תא שבו מצביע לראש רשימה מקושרת שמורכבת מתאים מטיפוס **DListNode**. כל רשימה במערך תכיל את כל המילים אשר מסתיימות באותה אות. לדוגמא, כל המילים אשר מסתיימות באות a יאוחסנו ברשימה אחת שאחד התאים במערך יצביע לראשה, וכל המילים אשר מסתיימות באות D יאוחסנו ברשימה אחרת שאחד התאים האחרים במערך יצביע לראשה.

על הפונקציה להוסיף את **freq** לכמות המופעים של המילה **word** במערך **word_lists_arr** אם המילה **word** קיימת במערך. יש לשים לב למקרים הבאים:

- אם המילה אינה קיימת ו-**freq>0**, יש להוסיף את המילה למערך במקום המתאים עם כמות המופעים **freq**. במידת הצורך יש להוסיף רשימה חדשה.
- אם המילה אינה קיימת ו-**freq<=0**, יש להדפיס על המסך את ההודעה

Error: word does not exist and freq<=0

- אם המילה קיימת ו-**freq<0** והוספת **freq** גורמת לכמות המופעים של המילה במערך להיות קטנה או שווה לאפס, יש למחוק את המילה מהמערך. אם זו המילה היחידה ברשימה, יש להסיר את הרשימה מהמערך ולכווץ את גודלו.

כל רשימת מילים צריכה להיות ממוינת בסדר לקסיקוגרפי יורד.
כמו-כן, המערך צריך להיות ממוין לפי האותיות שבהן מסתיימות המילים ברשימות.

שימו לב: אין להקצות מראש את המערך לכל האותיות האפשריות.

הפונקציה תחזיר את מספר הרשימות במערך **word_lists_arr** לאחר ההוספה.

דוגמא: אם נתחיל ממערך ריק ונשלח את המילים **hey 2, bye 3, When 3, Why 1** המערך יכיל 3 תאים. הרשימה בתא 0 תכיל את כל המילים אשר מסתיימות באות **e** כלומר את המילה **bye** עם מספר מופעים 3. הרשימה בתא 1 תכיל את כל המילים שמסתיימות ב-**n** כלומר, הרשימה תכיל את המילה **When** עם מספר מופעים 3. הרשימה בתא 2 תכיל בסדר לקסיקוגרפי

מורד את כל המילים שמסתיימות ב-y; המילה **hey** עם **frequency=2** ואחריה את המילה **Why** עם **frequency =1**.

אם כעת נקרא לפתקציה עם **bye -4** אזי הרשימה בתא 0 תמחק, המערך יכווץ ואחרי הכיווץ המערך יכיל בתאים 0 ו-1 את הרשימות שהיו לפני ההוספה בתאים 1 ו-2.

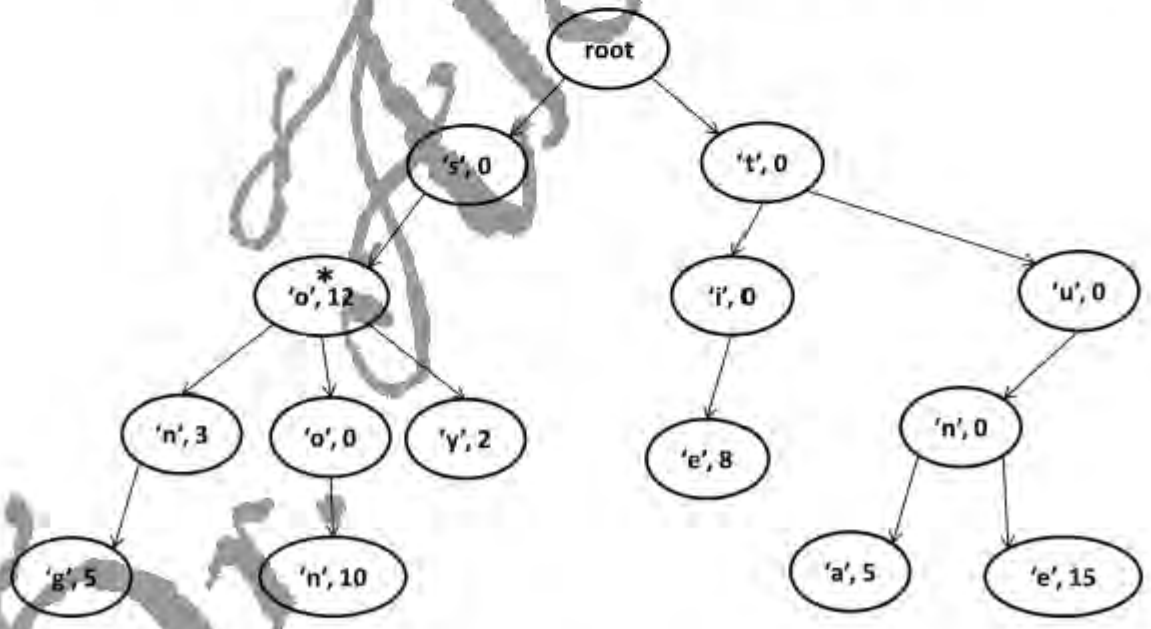
שאלה 2

נתונה ההגדרה הבאה של עץ לשמירת מילים ותדירותן:

```
typedef struct _WordTreeNode
{
     char ch;
     unsigned short frequency;
     struct _WordTreeNode **children;
} WordTreeNode;

Typdef struct _WordTree
{
     WordTreeNode *root;
} WordTree;
```

בכל צומת נשמרת אות **אחת**. המילים שנשמרות מתקבלות משרשור האותיות שבצמתים אשר במסלולי העץ החל מהשורש (לאו דווקא עד עלה). בכל צומת X ישנו גם משתנה אשר בו נשמרת תדירות הופעת המילה אשר אותיותיה מתחילות בשורש ומסתיימות בצומת X. לדוגמא, העץ הבא:



מייצג את המילים הבאות ותדירותן:

so 12, son 3, song 5, soon 10, soy 2, tie 8, tuna 5, tune 15

כל צומת שומר את המצביעים לילדיו במערך children כאשר המצביעים ממוינים בסדר עולה לפי האותיות שנשמרות בילדים. סוף המערך יסומן ע"י NULL.
לדוגמא, בתרשים לעיל, בצומת שמסומן ב- * : התא children[0] יכיל את המצביע לילד שמכיל את האות 'n', התא children[1] יכיל מצביע לילד שמכיל את האות 'o', והתא children[2] יכיל המצביע לילד שמכיל את האות 'y'. התא children[3] יכיל NULL.

כתבו את הפונקציה:

unsigned int InsertToWordTree (char *word, unsigned char freq, WordTree *tree)

הפונקציה מקבלת מחרוזת **word** שמכילה מילה אחת, תדירות **freq**, ועץ מילים **tree**. המילה מורכבת מהאותיות AB...Z והאותיות ab...z בלבד.
על הפונקציה להוסיף את המילה **word** ותדירותה לעץ **tree**. אם המילה מיוצגת כבר בעץ, יש להוסיף **freq** לשדה ה-**frequency** שלה. אחרת, יש לטפל במקרה שבו היא לא מיוצגת בעץ.
הפונקציה תחזיר את מספר המופעים המעודכן של **word** בעץ **tree**.

שאלה 3

כתבו את הפונקציה:

void ConvertWordListsArrToWordTree(DListNode **word_lists_arr, WordTree *tree)

הפונקציה מקבלת מערך של מילים ותדירותן כפי שהוגדר בשאלה 1. הפונקציה מקבלת by ref עץ לא מוקצה כפי שהוגדר בשאלה 2. על הפונקציה לבנות עץ מהמילים ותדירותן אשר ב- **word_lists_arr**.

שאלה 4

כתבו את הפונקציה:

void ReadFileToWordLists (char *fname, DListNode ***word_lists_arr)

הפונקציה מקבלת שם קובץ טקסט **fname** אשר מכיל צמדים של מילה ואחריה תדירות.
הפונקציה מקבלת **by ref** מערך של רשימות כפי שהוגדר בשאלה 1. על הפונקציה לקרוא את הצמדים מהקובץ ולבנות מערך של רשימות כפי שהוגדר בשאלה 1.
שימו לב: מילה יכולה להופיע יותר מפעם אחת עם תדירויות שונות.

שאלה 5

יש לכתוב את הפונקציה:

int WriteCompressedWordTree (WordTree tree, char *fname);

הפונקציה מקבלת:
tree – עץ של מילים כפי שהוגדר בשאלה 2.
fname – שם קובץ פלט

מעוניינים לכתוב את המידע שבעץ לקובץ בינארי באופן דחוס. המילים ותדירותן תיכתבנה זו אחר זו בסדר לקסיקוגרפי עולה של המילים.
הדחיסה מתאפשרת היות והמילים יכולות להכיל רק את האותיות AB...Z והאותיות ab...z ולכן מספיקים 6 ביטים כדי לקודד כל אות. האותיות תקודדנה באופן הבא:
האות a תקודד ע"י 0, האות b ע"י 1, z ע"י 25, A ע"י 26, B ע"י 27, וכן הלאה.
מבנה הקובץ יהיה כדלקמן:
בתחילה יהיה מספר מטיפוס unsigned short אשר יכיל את מספר המילים.
אח"כ תקודדנה המילים זו אחר זו באופן הבא:

מספר האותיות במילה יקודד ע"י 5 ביטים. אחריו תקודדנה האותיות זו אחר זו – 6 ביטים לכל אות. לאחר המילה, תקודד תדירותה לדוגמא, אם המערך מכיל רק את המילים ONE בתדירות 17 ו-three בתדירות 9, הקובץ הבינארי ייראה כך (הרווחים הם למטרת נוחות הקריאה ואינם מופיעים בקובץ):

$$\underbrace{00000000\ 00000010}_{2\ words}\ \underbrace{00011}_{3}\ \underbrace{101000}_{O}\ \underbrace{100111}_{N}\ \underbrace{011110}_{E}\ \underbrace{00000000\ 00010001}_{17}$$

$$\underbrace{00101}_{5}\ \underbrace{010011}_{t}\ \underbrace{000111}_{h}\ \underbrace{010001}_{r}\ \underbrace{000100}_{e}\ \underbrace{000100}_{e}\ \underbrace{00000000\ 00001001}_{9}\ 00$$

<u>שאלה 6</u>

יש לכתוב את הפונקציה:

int WriteCompressedFileFromTextFile(char *iname, char *oname);

הפונקציה מקבלת שם קובץ קלט טקסט **iname** אשר מכיל צמדים של מילה ואחריה תדירות. מילה יכולה להופיע יותר מפעם אחת עם תדירויות שונות.

על הפונקציה ליצור קובץ בשם **oname** כפי שהוגדר בשאלה 5 מהצמדים אשר מופיעים בקובץ **iname**. אם מילה מופיעה יותר מפעם אחת בקובץ **iname**, יש לכתוב אותה עם סך כל המופעים שלה בקובץ **oname**.

הנחיות כלליות

יש לתעד את התכניות.
יש להקפיד על יעילות וחסכון בזמן ריצה וזיכרון.
יש להקפיד שגודל פונקציה לא יחרוג ממסך אחד.
יש לפנות זיכרון שהוקצה דינמית ואשר אין בו צורך יותר.
יש לבדוק שפתיחת קובץ הצליחה.
יש לסגור קובץ לאחר סיום השימוש בו.
יש להשתמש ב#define היכן שנחוץ.
אין לחרוג מה-prototype-ים שהוגדרו אך ניתן ליצור פונקציות עזר לפי הצורך.