תכנות מערכות א'
מטלה 4
מבנים והקצאת זיכרון דינאמית

אחראי מטלה: יבגני הרשקוביץ נייטרמן
Neiterman21@gmail.com

שימו לב:

- המטלה היא בזוגות או יחידים – **לא שלשות**.
- עליכם לבצע את פקודת הקומפילציה עם הדגל -Wall על מנת לוודא שתוכניתכם מתקמפלת ללא אזהרות. תכנית שמתקמפלת עם אזהרות תגרור הורדת נקודות.
- עליכם לוודא שתוכניתכם מתקמפלת ורצה על גבי מערכת ההפעלה ubuntu עם קומפיילר gcc
- יש להגיש את המטלה ב-git . יש להגיש קובץ txt. השורה הראשונה תכלול את הכתובת של ה-git (לא הURL) מאתר ה-github. השורה השנייה תכלול את מזהה ה-commit הרלוונטי והשורה השלישית את תעודות הזהות של הסטודנטים המגישים מופרדים ברווח.
- הנכם נדרשים לקוד קריא ונקי.
- בתרגיל זה, אין כל הנחה על גודל הקלט ולכן יש לעבוד עם הקצאה דינאמית.
- בכל מקום בו יש צורך בשימוש בקבועים בעלי משמעות יש להגדיר אותם באמצעות define
- יש לבדוק זליגת זיכרון באמצעות valgrind (המדריך נמצא במודל). תוכנית אשר תהיה בה זליגת זיכרון לא תזכה לניקוד מלא.

בתרגיל זה אנו נממש עץ ונאחסן בו טקסט.

<u>עצה:</u> אל תתחילו לעשות דבר בתרגיל-הבית לפני שקראתם את התרגיל בעיון לפחות פעם אחת מההתחלה ועד הסוף. חלוקה נכונה של הבעיה לתת-בעיות קטנות יותר, תקל עליכם הן בהגדרת הפונקציות שלכם והן בפתרון מלא ונכון של הבעיה תוך מימוש פתרונכם בשפת C.

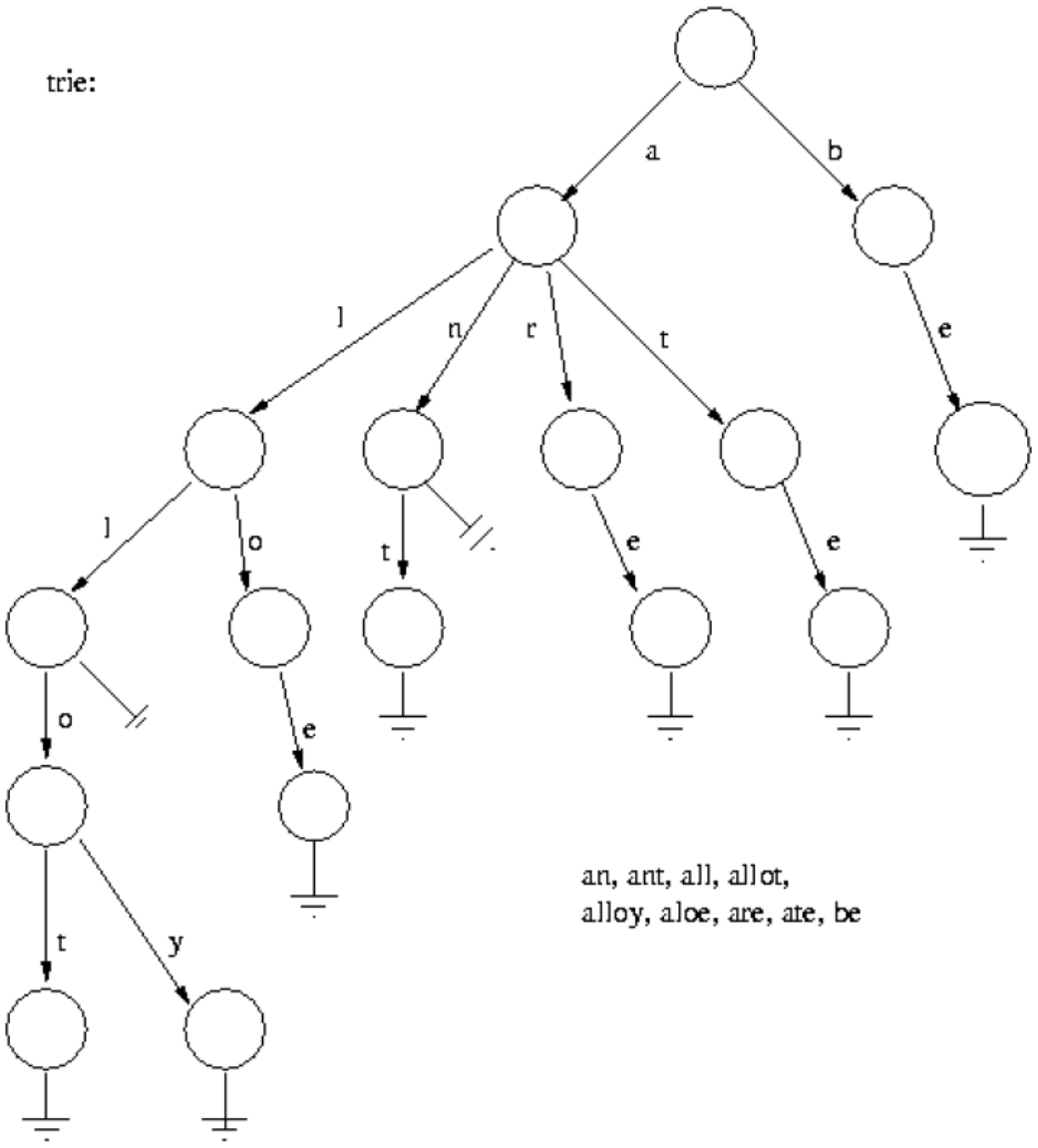
מוטיבציה:

יישומים רבים לניתוח אוטומטי של טקסטים עושים שימוש בנתונים סטטיסטיים הנאספים מתוך הטקסטים. דוגמא לנתון סטטיסטי כזה הוא מספר המופעים של כל מילה בטקסט נתון, שכן מסתבר כי ניתן להסיק מסקנות רבות משכיחותה של מילה כזאת או אחרת.
בתרגיל זה אנו נעשה שימוש במבנה הנתונים TRIE כדי לאחסן את המילים אשר נראו בטקסט. כמו-כן נשמור עבור כל מילה במבנה הנתונים את מספר המופעים שלה בטקסט הנקרא.
לTRIE יש יתרון באחסון של אוספי מילים. בפרט ניתן לאחסן ביעילות טובה מאוד מילים בעלות רישא משותף. עליכם יהיה להגדיר את המבנים של צומת בעץ (ואולי גם מבנים נוספים), וכן עליכם לממש את פעולות יצירת העץ והריסתו, גידול העץ על-ידי הכנסת מילים, תחזוק מוני תדירות של מילים וכיוצא באלה.

TRIE:

נגדיר את מבנה הנתונים TRIE. הTRIE הוא עץ בו מאחסנים מחרוזות. נתבונן בדוגמא הבאה בה אנו מאחסנים את המילים: an, ant, all, alloy, aloe, are, ate, be

ה TRIEשיכיל את המילים יראה כך:



המשימה:

עליכם לכתוב תכנית אשר תקרא מילים מתוך הקלט הסטנדרטי ותדפיס את רשימת המילים אשר נראו בקלט עם מספר המופעים של כל מילה ומילה. את הרשימה יש להדפיס, כתלות בפרמטר משורת הפקודה, באחד מן האופנים הבאים:

- רשימה ממוינת לפי סדר לקסיקוגרפי עולה (ברירת המחדל)
- רשימה ממוינת לפי סדר לקסיקוגרפי יורד (לפי פרמטר).

הגדרת הרצת התוכנית

./frequency

תדפיס את רשימת המילים שנראו בקלט יחד עם מספר המופעים לכל מילה ממוינת בסדר לקסיקוגרפי עולה.

./frequency r

תדפיס את רשימת המילים שנראו בקלט יחד עם מספר המופעים לכל מילה ממוינת בסדר לקסיקוגרפי יורד.

דגשים במימוש:

אנו נתמוך רק באותיות קטנות לכן עליכם לדאוג להמרה של אותיות גדולות לאותיות קטנות.
עליכם להתעלם מכל סימן בקלט שאינו אחת מעשרים ושש האותיות הקטנות באנגלית.
את הפלט עליכם להדפיס באופן הבא: כל מילה מופיעה בשורה נפרדת. מילה ומספר המופעים שלה יופרדו ע"י רווח בודד. למשל, עבור הקלט

abc bac bac abc ddd aaa.

ועבור הרצה ללא פרמטרים בשורת הפקודה, נצפה לפלט הבא:

aaa 1
abc 2
bac 2
ddd 1

זכרו לשחרר זיכרון שהוקצה על-ידכם באופן דינאמי לאחר שסיימתם את השימוש בזיכרון.

הבדיקה תתבצע עם valgrind שיחפש זליגת זיכרון (זיכרון שלא שוחרר).

הצעת הגדרות לשימושכם (אינכם חייבים להשתמש בהם):

```
#define NUM_LETTERS ((int)26)

typedef enum {FALSE=0, TRUE=1} boolean;

typedef struct node {
char letter;
long unsigned int count;
struct node* children[NUM_LETTERS];
} node;
```

צומת יכיל את

- האות המופיעה על הקשת הנכנסת אליו
- מונה לספירת מספר מילים
- מערך בן עשרים ושישה מצביעים לצומת, עבור ילדים פוטנציאליים

אתם רשאים להוסיף שדות נוספים, אם תרצו, או להשתמש במבנים אחרים לפי הבנתכם. חישבו למשל, כיצד תדעו האם לצומת יש או אין ילדים. חישבו כיצד תדעו האם צומת מסמן גם סוף של

מילה. בכל מקרה, כאשר אתם מגדירים את הצומת שלכם, על שדותיו, הקפידו לתעד כל הנחה וכל החלטה שאתם מקבלים.

<u>זכרו:</u> כל ערך קבוע אשר אתם עושים בו שימוש, רצוי להגדירו במקום מסודר בתחילת התוכנית (ע"י #define).

הקפידו על ניהול זיכרון נכון (הקצאה ושחרור).

הגשה

במודל יש להגיש קובץ txt בשם hw4.txt
הקובץ יכיל 3 שורות. קישור ל-git שלכם, מספר commit ותעודות זהות המגישים מופרדים ברווח.

על הגיט שלכם להכיל makefile ברמה הראשונה שלו (לא בתתי תיקיות) ולאחר הרצת הפקודה make all עליכם לקמפל וליצור את תוכנית ההרצה שתקרא frequency
הקלט לתוכנית יועבר בצודה של redirection כמו במטלות הקודמות. אין צורך בעבודה עם קבצים (שלא נלמדו בזמן פרסום המטלה).

יועלה למודל סקריפט בדיקת הגשה, אשר יבדוק את נכונות קובץ ההגשה שלכם ואת נכונות הגיט שלכם (בדיקה פשוטה) לצורך הבדיקות האוטומטיות שלנו. יש לבדוק את קובץ ההגשה שלכם ונכונות הגיט לפני הגשת התרגיל.

יש לפתוח תיקייה חדשה ולהכניס לתוכה את קובץ הבדיקות, קובץ ההגשה וקובץ דוגמת הקלט שיעלו למודל. (הקבצים היחידים בתיקייה) ולהריץ את השורה הבאה.

```
python check_submition.py hw4.txt
```

אם התוכנית והגיט שלכם תקינים יודפסו תעודות הזהות של המגישים. תופיע לכם תיקיית גיט חדשה, היא תתקמפל והתוכנית שהגשתם תורץ עם קלטים פשוטים ופלט התוכנית שלכם יודפס למסך.

הסקריפט עולה בגרסת פייטון 2.7 אם אתם עובדים על לינוקס מלא כנראה שיש לכם גרסה 3. נסו להריץ :

```
python2.7 check_submition.py hw4.txt
```

יש לוודא שהתוכנית מתקמפלת ורצה על גבי ubuntu עם gcc. אתם רשאים לעבוד עם כל עורך קוד שנראה לכם לנכון **אך סביבת הבדיקה תהיה ubuntu**