תרגיל 4–מחרוזות, מצביעים, הקצאה דינמית ורקורסיה

להגשה עד ה- 11.01 בשעה 23:50

הוראות ההגשה ודגשים מיוחדים **מופיעים בסוף התרגיל! חובה לקרוא!**

מי שלא יעבוד בדיוק לפי ההנחיות יפסיד נקודות רבות מציון התרגיל! ראו הוזהרתם!

חלק א' – תאורטי 20% (מענה בקובץ טקסט מוקלד בלבד):

משימה 1 (10%) – מה עושה התוכנית?

בכל סעיף עליכם להסביר שורה שורה מה מתבצע בתוכנית וכמובן לפרט **ולהסביר** מהו הפלט הסופי של התוכנית. (תשובה ללא פירוט והסבר לא תתקבל).

א.

```
#include<iostream>
int main() {
    int array[2][2][2] = {10, 2, 7, 4, 1, 30, 5, 8};
    int *p, *p2;
    p = &array[1][1][1];
    p2 = (int*) array;
    std::cout << *p<<" "<< *p2;
    return 0;
}
```

ב. (בהנחה שהתא הראשון במערך נשמר בכתובת 1000 בזיכרון).

```
#include<iostream>
int main(){
    int a[3][4] = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12 };
    std::cout << a[0]+1 <<" "<<*(a[0]+1)<<" "<<*(*(a+0)+1);
    return 0;
}
```

משימה 2 (10%) – מהי השגיאה?

בכל סעיף הסבירו איזה סוג שגיאה יש בקוד הנ"ל (קומפילציה, קישור, ריצה, לוגית או אין שגיאה) וכן הסבירו ממה בדיוק היא נגרמת ואיך (אם בכלל) ניתן לתקנה.

א.

```
#include<iostream>
int main() {
    int *p;
    *p=100;
    return 0;
}
```

ב.

```
#include<iostream>
#define SIZE 5
int main() {
    int i;
    int *p = new int[SIZE];
    for(i=0; i<SIZE; ++i) {
         std::cin >>*p;
         p++;
    for(i=0; i<SIZE; ++i)
         std::cout<<p[i]<<"\t";
    delete []p;
    return 0;
}
```

ג.

```
#include<iostream>
int main() {
    int array[] = {10, 20, 30, 40, 50};
    for(int j=0; j<5; ++j) {
        std::cout<<array<<std::endl;
        array++;
    }
    return 0;
}
```

ד.

```
#include<iostream>
#define SIZE 3
int main() {
    int arr[SIZE] = {1,2,3,4,5};
    for(int i=0; i<SIZE; ++i)
         std::cout<<arr[i]<<" ";
    return 0;
}
```

ה.

```
#include<iostream>
int main() {
    char str1[] = "Hello World";
    char str2[] = "Bye Bye";
    char *ptr1 = str1;
    char *ptr2 = str2;
    if(ptr1>ptr2)
        std::cout<<ptr1<<" is bigger than "<< ptr2;
    else
        std::cout<< ptr2<<" is bigger than "<< ptr1;
    return 0;
}
```

חלק ב' – מעשי 80% (מענה בקובץ cpp. בלבד!):

בתרגיל זה עליכם לממש את כל מה שכתוב בתרגיל. שימו לב כי מדובר בתוכנית אחת(!) - פונקציה main אחת – שמפעילה הרבה פונקציות אחרות! **חישבו היטב כיצד לחלק את התוכנית לפונקציות!**

המערכת:

בתרגיל זה אתם מתבקשים לבנות מילון.

מילון הוא מבנה נתונים ממוין (לפי סדר ה- A, B, C) המכיל מילים באנגלית ולכל מילה ישנה גם הגדרה (הגדרה היא משפט באנגלית המסביר את פירושה של המילה).

המילון עצמו ישמר כמצביע ברמה שלישית ל-char (***char). הרעיון הוא שהמצביע בעצם יוקצה דינמית למערך של "כניסות" במילון. וכל "כניסה" (מטיפוס **char) – תהיה בעצם מערך דינמי בגודל 2 של מחרוזות שבתא הראשון שלו (מסוג *char) תהיה המילה עצמה ובשני ההגדרה של אותה מילה.

בתחילת התוכנית המערכת תקבל את הנתונים שיכיל המילון מהמשתמש, תבנה את המילון לפי המידע ואז תיכנס לשלב החיפוש. בשלב החיפוש המשתמש יכול לחפש מילים במילון ולקבל את המשמעות שלהם. כאשר המשתמש מחליט כי אינו רוצה לחפש יותר במילון הוא יקליד את בקשת היציאה ואז המערכת תנקה ותשחרר את הזיכרון בצורה מסודרת(!) לפני היציאה.

פירוט השלבים

שלב קבלת הנתונים (25%):

בהתחלה לא ידוע למערכת כמה מילים יוכנסו למילון ולכן יש במערכת מצביע מסוג ***char.
שלב ראשוני יהיה לקבל מהמשתמש את מספר המילים שהוא מעוניין להכניס למילון כקלט של מספר שלם. בהתאם לקלט המערכת תיצור את המערך הראשוני (הקצאה דינמית!).

כעת יש לקלוט מהמשתמש כל "כניסה" במילון בפני עצמה.
בתחילת כל "כניסה", התא הנוכחי של המערך (שהוא מצביע מסוג **char), תוקצה למערך בגודל 2 של מחרוזות (להחזקת מילה + ההגדרה שלה).
ואז בתא הראשון של המערך החדש (כל תא מסוג *char) תוכנס המילה ובשני תוכנס ההגדרה.

המשתמש יזין את הנתונים כך:

- המילה עצמה תוזן כרצף של לא יותר מ-80 תווים שלא כולל בתוכו רווחים ולאחר מכן יקיש המשתמש אנטר (ENTER).
- הגדרות של המילה היא רצף של לא יותר מ-200 תווים שכן עלול להכיל רווחים, ובסוף ENTER.

יש לוודא שאחסון המילים וההגדרות יהיה יעיל מבחינת מקום (זאת אומרת שיוקצה בדיוק הגודל הדרוש ולא יהיה בזבוז).

<u>שלב בניית המילון (25%):</u>

לאחר שהמשתמש הכניס את כל המילים המערכת רוצה לבנות את המילון. לשם כך יש:

1. לתקן את הכתיב (10%):

a. במילים – כל מילה תתוקן כך שהאות הראשונה שלה תהיה uppercase וכל שאר האותיות יהיו lowercase. **בונוס (5%) שהפונקציה המתקנת את המילה תהיה רקורסיבית!**

b. בהגדרות - כך שהאות הראשונה במשפט וכן אות ראשונה בכל מילה המופיעה לאחר נקודה (זאת אומרת שיש נקודה, רווח ואז את התו שיש לשנות ל-uppercase) תתחיל באות uppercase וכל שאר האותיות בהגדרה יהיו lowercase. **גם כאן בונוס (5%) על שימוש בפונקציה רקורסיבית!**

שימו לב: מובטח לכם כי כל תו המתחיל מילה או משפט יהיה באמת תו שהוא אות באלף-בית האנגלי. לגבי שאר התווים ייתכן שיהיו אותיות (ואז יש לוודא שהם lowercase) וייתכן שיהיו תווים אחרים – ואז אין לבצע בהם דבר.

2. למיין את כל המילים בסדר לקסיקוגרפי (10%) (A, B, C…).

שימו לב למיין את המילון בצורה היעילה ביותר האפשרית <u>וחובה</u> לעשות זאת תוך שימוש באלגוריתם רקורסיבי.

3. (5%) להיפטר ממילים זהות המופיעות יותר מפעם אחת (אפילו אם ההגדרות שלהם שונות!) יש לקחת רק את המופע הראשון של מילה (הפעם הראשונה שבה הוכנסה).

לאחר המיון והצמצום של המילים הכפולות – יש לוודא שוב כי הקצאת המילון לא מבזבזת מקום במערכת (זאת אומרת שאם קודם הוקצו יותר תאים במערך הראשי משהיה צריך כי חלק מהמילים היו כפולות – עכשיו צריך לתקן זאת...)

שלב החיפוש (20%):

בשלב זה המשתמש יכניס למערכת מילים שהוא מעניין לחפש במילון והמערכת תדפיס תוצאות בהתאם:

- למילה הנמצאת במילון – תודפס למסך ההגדרה שלה.
- למילה שלא נמצאת במילון – יודפס למסך "Unknown word!"

המשתמש יוכל להמשיך להכניס מילים כרצונו, עד שיכניס את המילה "exit" שמסמנת כי המשתמש רוצה לצאת מהמערכת (מובטח כי exit לא תהיה אחת המילים במילון).

יש לוודא כי חיפוש המילה במילון הוא יחסית יעיל (זאת אומרת שמשום שהמילון ממוין בהכרח אין חובה לחפש בכל המילון, אלא ניתן לחפש בדילוגים תוך מעבר חלקי בלבד על המערך) **שימו לב חובה לממש את החיפוש תוך שימוש באלגוריתם רקורסיבי.**

שלב היציאה (10%):

חובה לוודא יציאה מסודרת תוך שחרור כל הזיכרון המוקצה דינמית!

הערות:

1. שימו לב שכאשר הקלט מהמשתמש הוא מסוג מסוים או מבנה מסוים, אתם יכולים להניח שאנו אכן נכניס את הסוג\המבנה המתאים. אך לא בהכרח ערכים חוקיים (אינדקסים חוקיים) – באחריותכם לוודא את חוקיות הקלט!
2. אחרי כל הדפסה יש לבצע ירידת שורה.
3. **שימוש בפונקציות רגילות במקום בו נדרשתם להשתמש בפונקציות רקורסיביות יוביל לקבלת אפס על חלק זה בתרגיל – ראו הוזהרתם!**
4. יש להקפיד על תכנות נכון:
   a. כל הערכים שהם קבועים חייבים להיות מוגדרים כ: define, const או enum, בהתאם לצורך, אין זה נכון להשתמש בקבועים מספריים.
   b. יש לרשום הערות בשפע! **ובאנגלית בלבד**.
   c. יש לנסות ולייעל את הקוד והתוכנית ככל שניתן ולהשתמש באלגוריתמים יעילים כל הניתן!
   d. לפני כל בקשת קלט יש להדפיס למשתמש הוראה איזה קלט מבוקש.
   e. יש להקפיד לבצע חלוקה לפונקציות. אין לכתוב קוד כפול!
   f. בהקצאה דינמית – חובה לעבוד בצורה מסודרת – לא לאבד את המצביע ולשחרר בסוף את כל מה שהוקצה!
   g. יש להקפיד על הזחות, וכיתוב נכון וקריא.
   h. יש להקפיד על כל כללי התכנות הנכון כפי שנלמדו בכיתה.
5. בהצלחה ☺

הנחיות הגשה

- תרגילים הם ביחידים! כל עבודה משותפת (כולל העזרות בקוד של מתגבר או בקוד כתוב מהאינטרנט) אסורה ותיענש בחומרה!

- ההגשה היא של שני קבצים בלבד: קובץ ה**קוד** (כקובץ cpp.) וקובץ מוקלד (בלבד) לחלק התאורטי. הקבצים ישלחו **אך ורק** דרך המערכת!
שליחת קבצים אחרים – תוביל לאיפוס ציון התרגיל. שליחת קבצים מיותרים תוביל להפחתת נקודות.

- כל הקוד נכתב בקובץ אחד, אך יחולק להרבה פונקציות. שימו לב לכתוב את הצהרות הפונקציות מעל ה-main ואת המימוש מתחת.
- ההגשה של קובץ הקוד היא הגשה של קובץ מוכן לקימפול והרצה! הבודק לא ישנה דבר בקובץ לפני בדיקתו. כל מי שיגיש קובץ שחלקו בהערה – החלק הנ"ל לא ייבדק לו. כל מי שיגיש קובץ עם שגיאות – ציונו יהיה בהתאם. דאגו לבדוק את הקבצים **לפני השליחה!!!**

- בתחילת קובץ הקוד חובה להוסיף את התיעוד הבא:

```
/* Assignment: 4
        Author: Israel Israeli, ID: 01234567
*/
```

כמובן שיש לעדכן את השמות ומספרי תעודות הזהות שלכם.
- כמו כן חובה לציין את שמכם המלא ומספר ת"ז שלכם בתחילת הקובץ המוקלד.

- הארכות יינתנו אך ורק במקרים חריגים (מילואים, אבל על קרובים ומחלה חריפה!) ובצרוף אישורים מתאימים! כמו כן חובה ליצור קשר עם המרצה האחראית על התרגיל לפחות יומיים לפני חלוף הדד-ליין!

- ההגשה היא עד התאריך האחרון לתרגיל: 11.01 בשעה 23:55. הגשה מאוחרת לא תתקבל. קחו זאת בחשבון ותכננו את זמנכם בהתאם!