תכנות מונחה עצמים
עבודת הגשה 3

מועד הגשה: 28.4.2019 בשעה 23:50

הוראות הגשה:

1. **אנא קראו בעיון את כל תיאור העבודה בטרם תתחילו לכתוב קוד.**
2. הגשה באופן עצמאי בלבד. הגשה בקבוצות תוביל לציון 0 בעבודה.
3. אין לשתף או להעתיק את העבודה או חלקים ממנה. עבירה על הוראה זו תוביל לציון 0 בעבודה.
4. הגשה דרך מערכת מודול בלבד. שום עבודה לא מתקבלת במייל!
5. למחלקות המפורטות בעבודה ניתן להוסיף מתודות נוספות אך הם חייבות להיות private.
6. יש למקם כל מחלקה שיהיה עליכם ליצור, בשני קבצים נפרדים H ו-CPP. יש להכניס את החלק התיאורטי בקובץ וורד נפרד. יש להכניס את כל הקבצים של חלק ב' לתיקיה בשם Ex ואז לכווץ יחד עם קובץ וורד. נדרש להגיש קובץ אחד בפורמט RAR או ZIP המכיל את כל הקבצים של כל השאלות. לקובץ המכווץ יהיה שם המהווה את מספר ת.ז. של המגיש.
7. **שאלות ובקשות בקשר לעבודה להפנות אך ורק למרצה סבטלנה רוסין, במייל: sceassign2016@gmail.com.**

חלק א' – תאורטי (מענה בקובץ טקסט – וורד): 8 נקודות

משימה 1:

1) מה היא המוטיבציה להעמסת אופרטור כפונקציית friend ? יש לתת דוגמה.
2) מהי האסטרטגיה בהעמסת אופרטור = (השמה)? למה הוא מתנת קומפיילר לא טובה ?
3) איך ניתן להבדיל בין העמסת אופרטורים postfix ו-prefix.
4) האם פונקציה friend יכולה להיות const ?
5) אם נרצה לתמוך באופרטור [ ] משני צידי ההשמה איך נגדיר אותו . יש לתת דוגמה . למשל,

```
class String{ char* data; int length; ……};

String  objStr(….);
char letter = objStr[1];
objStr[2]='B';
```

6) איך ניתן להמיר את הטיפוס הסטנדרטי (int , float,…) לסוג האובייקט (user-defined type)? ואיך להפך ? יש לתת דוגמאות קצרות.

משימה 2:

1) האם חלק מהקוד הבא הינו תקין : אם כן יש לתת דוגמאות ההרצה (בדרייבר), אם לא – הסבר מדוע.

```
class Point{
        int x,y;
public:
        ..........
        Point& operator+=(const Point& );
        Point& operator+=(int );
};
Point& Point::operator+=(const Point& other){
                x+=other.x;
                this->y+=other.y;
                return * this;
        }
Point& Point::operator+=(int num){
                this->x+=num;
                y+=num;
                return * this;
}
```

2) האם חלק מהקוד הבא הינו תקין : אם כן יש לתת דוגמאות ההרצה (בדרייבר), אם לא – כיצד יש לתקן אותו.

```
class Student{
        int id;
        char* name;
public:
        ..........
        Student (const Student& );
        Student& operator=(const Student& );
};
Student::Student(const Student& other ){
        *this=other;
}
Student& Student::operator=(const Student& other ){
        if(this != & other){
                id=other.id;
                delete[] name;
                name=new char[strlen(other.name)+1];
                strcpy(name,other.name);
        }
        return *this;
}
```


המכללה האקדמית להנדסה סמי שמעון
קמפוס באר שבע ביאליק פינת בזל 84100 | קמפוס אשדוד ז'בוטינסקי 84, 77245 | www.sce.ac.il | חייג: *מהנייד

חלק ב' – מעשי (ההגשה היא של קבצי ה- CPP ו-H בלבד) 92 נקודות:

משימה מס' 1 (70 נקודות):

חל איסור להשתמש בכל הכלים המוכנים עבור מחרוזת (cstring) ב C/ ++C. כלומר אסור להשתמש בספריה string.h (אפילו שהיא build-in) ובספריה (string) המייצגת מחלקת מחרוזת של ++C.

המחלקה שלכם String צריכה לייצג מחרוזת וכל הפונקציונליות שלה.

המערכת:

בתרגיל זה אתם מתבקשים לבנות **מילון**.

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

לדוגמא ההגדרה המילונית (לפי מילון Merriam-Webster) של המילה Dictionary הינה:

1
: a reference source in print or electronic form containing words usually alphabetically arranged along with information about their forms,pronunciations, functions, etymologies, meanings, and syntactical and idiomatic uses.
2
: a reference book listing alphabetically terms or names important to a particular subject or activity along with discussion of their meanings and applications.
3
: a reference book listing alphabetically the words of one language and showing their meanings or translations in another language.
4
: a computerized list (as of items of data or words) used for reference (as for information retrieval or word processing).

ראה את הדוגמה של המילון Merriam-Webster עבור המילה integrity:

http://www.merriam-webster.com/dictionary/integrity

לצורך ביצוע משימה אתם מתבקשים לבנות את המחלקות הבאות:

1. מחלקה ***String*** המתארת מחרוזת דינאמית (אוסף מסוג char, הכוללת אורך המחרוזת).

המחרוזת יכולה להכיל משפטים כאשר תו '.' יפריד בין המשפטים .

יש להגדיר בה את כל הבנאים הדרושים, הורס, אופרטורים:

= (השמה) **-** בין האובייקטים מאותו סוג אבל אם תידרש לבצע השמה נוספת כמו בין האובייקט-מחרוזת ולבין מערך תווים אז להוסיף גם אותה.

== (בדיקת שיוון) – בדיקה מדויקת ז"א case sensitive.

=! (בדיקת אי שיוון)

>> (הפלט)

המכללה האקדמית להנדסה סמי שמעון
קמפוס באר שבע ביאליק פינת בזל 84100 | קמפוס אשדוד ז'בוטינסקי 84, 77245 | www.sce.ac.il | חייג: *מהפלפון

<< (הקלט)

=- (מוריד תו מהמחרוזת) - מקבל תו ספציפי , למשל 'A' ומוריד <u>את כל ההופעות שלו</u> מהמחרוזת במידה והתו קיים.

=+ (מוסיף תו למחרוזת) - מקבל תו ספציפי , למשל 'A' ומוסיף אותו לסוף המחרוזת הקיימת.

אופרטור [ ] המחזיר תו (יש לממש עם אופציית עדכון - ראה את חלק א' משימה 1 השאלה 5).

בנוסף יש לספק למחלקה (ולהפעיל במקומות הרלוונטיים) פונקציות ל"סידור" הנתונים:

לצמצם את כל הרווחים המיותרים (ז"א ההפרדה בין המילים תמיד תהיה ע"י רווח בודד בלבד, ליד הפיסוק (. , :) לא יהיו רווחים כלל .

כל משפט יתחיל באות גדולה , התווים האלפביתיים בתוך המשפט יהיו אותיות קטנות וכמובן יש להקצות זיכרון בהתאם לכך.

2. מחלקה *Definition* המתארת מושג במילון (הגדרה אחת).המחלקה אמורה לשמור את המילה עצמה (אובייקט של *String*) ופירושיה ז"א : מספר הפירושים (מספר שלם חיובי) ומצביע ל-*String* (רצוי מצביע כפול) . בנוסף לבנאים הדרושים יש להוסיף הורס ואופרטורים :

= (השמה) - בין האובייקטים מאותו סוג אבל אם תידרש לבצע השמה נוספת אז להוסיף גם אותה.

== בדיקת שוויון על פי התוכן המילה בלבד ללא התייחסות לפירושיה

>> (הפלט)

<< (הקלט)

=- מקבל מספר סידורי ומוריד פירוש אחד מההגדרה על-פי המספר הסידורי שלו בהגדרה

=+ מקבל פירוש המיוצג על-ידי אובייקט מסוג String ובמידה והפירוש לא מופיעה ברשימת הפירושים מוסיף אותו להגדרה, אחרת לא לנקוט באף פעולה נוספת. יש להוסיף את הפירוש החדש לסוף האוסף.

אופרטור **[ ]** מקבל את מיקום הפירוש בהגדרה ומחזיר את הפירוש הממוקם במיקום זה (מחזיר את פירוש ספציפי יחיד לפי מיקומו מתוך האוסף , יש לממש עם אופציית עדכון - ראה את חלק א' משימה 1 השאלה 5).

3. מחלקה ***Dictionary*** המתארת מילון המכיל הגדרות, ז"א : מספר ההגדרות (מספר שלם חיובי) ומצביע ל- *Definition* (רצוי מצביע כפול) . בנוסף לבנאים הדרושים יש להוסיף הורס ואופרטורים

= (השמה) **-** בין האובייקטים מאותו סוג אבל אם תידרש לבצע השמה נוספת אז להוסיף גם אותה.

== בדיקת שוויון בין 2 מילונים האם הם מכילים את אותם ההגדרות

>> (הפלט)

<< (הקלט)

=- מקבל מספר סידורי של הגדרה ספציפית ומוריד אותה מהמילון.

המכללה האקדמית להנדסה סמי שמעון
קמפוס באר שבע ביאליק פינת בזל 84100 | קמפוס אשדוד ז'בוטינסקי 84, 77245 | www.sce.ac.il | חייג: *מהפלאפון

+= מקבל הגדרה המיוצגת על-ידי אובייקט Definition ובמידה וההגדרה אינה קיימת במילון האופרטור יוסיף אותה לסוף האוסף, אחרת לא לנקוט באף פעולה נוספת.

אופרטור **[ ]** מקבל את מיקום ההגדרה במילון ומחזיר את ההגדרה הממוקמת במיקום זה.מחזיר את ההגדרה אחת ספציפית מתוך האוסף ,יש לממש עם אופציית עדכון- ראה את חלק א' משימה 1 השאלה 5.

4. מחלקה ***Menu*** המתארת תפריט. מדובר במחלקה שתנהל לנו את המערכת.

המחלקה אמורה לשמור את המילון (מצביע ל-*Dictionary*).

בנוסף לבנאים הדרושים והורס יש להוסיף אליה הפונקציה (מתודה) *MainMenu* . היא תדפיס למשתמש את כל האופציות הנתונות בפניו בתפעול המערכת ותיתן לו היכולת לבחור מה להפעיל . המתודה תיתן למשתמש את האפשרויות הבאות :

משתמש מקיש :	שם הפעולה :	הערות על הפעולה :
1	יצירת מילון והכנסת ערכים	לאחר הקשה, המערכת תקבל מהמשתמש את כלל הנתונים הדרושים (ההסבר בהמשך) ותבנה אובייקט-מילון. לאחר הרצה ראשונית של אופציה 1, לא יתאפשר למשתמש להריץ את אופציה זו בשנית.
2	הכנסת הגדרה חדשה למילון	לאחר הקשה, המשתמש יתבקש להזין את פרטי ההגדרה החדשה (מילה ופירושיה).
3	הוספת פירוש חדש להגדרה קיימת במילון	לאחר הקשה, המשתמש יתבקש להזין מילה ובמידה והיא קיימת במילון, הוא יוכל להוסיף עבורה פירוש חדש (ללא חזרות).
4	חיפוש הגדרה במילון	לאחר הקשה, המערכת תבקש להכניס מילה , תחפש ותדפיס את כל פירושיה או תציג הודעה שההגדרה לא מופיעה במילון .
5	הצגה של כל ההגדרות שיש להם לפחות פירוש אחד משותף	לאחר הקשה הזאת המערכת תציג את כל ההגדרות שבהם מופיעה לפחות פירוש אחד משותף.
6	יציאה	

ה-main

מייצרת אובייקט מסוג תפריט ומריצה את המתודה *MainMenu* וזהו!

בכל המחלקות הנ"ל אסור להוסיף שדות נוספים, אבל ניתן להוסיף מתודות ואופרטורים.

פירוט השלבים

אופציה 1 בתפריט הראשי:

המכללה האקדמית להנדסה סמי שמעון
קמפוס באר שבע ביאליק פינת בזל 84100 | קמפוס אשדוד ז'בוטינסקי 84, 77245 | www.sce.ac.il | חייג: *מהפלפון

1) שלב ראשוני יהיה לקבל מהמשתמש את מספר המילים (ההגדרות) שהוא מעוניין להכניס למילון כקלט של מספר שלם חיובי. בהתאם לקלט המערכת תבצע את הפעולות הרלוונטיות ותעבור לשלב קבלת ההגדרות .

2) שלב קבלת ההגדרות :

כעת יש לקלוט מהמשתמש כל הגדרה במילון בפני עצמה.

בתחילת כל הגדרה יזין המשתמש כמה הגדרות תשתייכנה למילה הנוכחית. בהתאם לכך המערכת תבצע את הפעולות הרלוונטיות.

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).
- כעת יקיש בזו אחר זו את ההגדרות של המילה (רצף של לא יותר מ-200 תווים שכן עלול להכיל רווחים) ובסוף כל ההגדרה תהיה ירידת שורה (ENTER).

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

יש לשים לב ,שהמשתמש לא חייב להכניס את ההגדרות למילון בצורה ממוינת (על פי המילים) לכן יש לחשוב כיצד ניתן ליצור את התהליך היעיל ביותר ליצירת הסדר הלקסיקוגרפי (A, B, C...) במילון.

3) עם קבלת הנתונים יש לסדר אותם לפי הדרישות הבאות בטרם יוטמעו במילון:

a. במילים – כל מילה תתוקן כך שהאות הראשונה שלה תהיה uppercase וכל שאר האותיות יהיו lowercase.

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.

c. 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.

d. להיפטר מרווחים מיותרים – *ז"א בין המילים יפיע רק רווח אחד בלבד או סימן* (כגון: פסיק, נקודה..)

e. לא להכניס למילון מילים שכבר קיימות בו (אפילו אם ההגדרות שלהם שונות! יש להיעזר באופרטור המתאים !) – במקרה של הכנסה של מילה קיימת יש לתת הודעה מתאימה ולא להכניס.

f. יש להיפטר מפרושים החוזרים על עצמם לאותה מילה (כלומר שלאותה מילה לא יהיו פירושים כפולים).

שימו לב כי הצורה בה יוטמעו ההגדרות והפירושים במילון באופציות הבאות זהה לתיאור של אופציה 1.

המכללה האקדמית להנדסה סמי שמעון
קמפוס באר שבע ביאליק פינת בזל 84100 | קמפוס אשדוד ז'בוטינסקי 84, 77245 | www.sce.ac.il | חייג: *מהנייד

אופציה 4 בתפריט הראשי:

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

- למילה הנמצאת במילון – יודפס למסך כמה הגדרות יש למילה ואז יודפסו ההגדרות השונות בצורה ממוספרת.
- למילה שלא נמצאת במילון – יודפס למסך "Unknown word!"

אופציה 5 בתפריט הראשי:

בשלב זה המערכת תמצא ותדפיס את כל ההגדרות הבעלות לפחות פירוש אחד משותף, ויש להדפיס את הרשימה בחלוקה לקבוצות הרלוונטיות (הקבוצה תהי בעלת לפחות 2 מילים "דומות") מאחר ויתכן כי ישנם מספר קבוצות שונות העונות על תנאי זה, במידה ואין אף קבוצה מתאימה יש להדפיס הודעה בהתאם.

הערה: לטובת התרגיל יש לחפש פירושים זהים , אך בפועל מצב זה נדיר.

אופציה 6 בתפריט הראשי:

חובה לוודא יציאה מסודרת תוך שחרור כל הזיכרון המוקצה דינמית!

הערה: בעבודה הזאת נכון לעכשיו לא נוצלו כל האופרטורים שהוגדרו – הם ימומשו בעתיד לצורך הרחבת המערכת. יש להשתמש בכל האופרטורים שהגדרתם בכל מקום שאפשר.

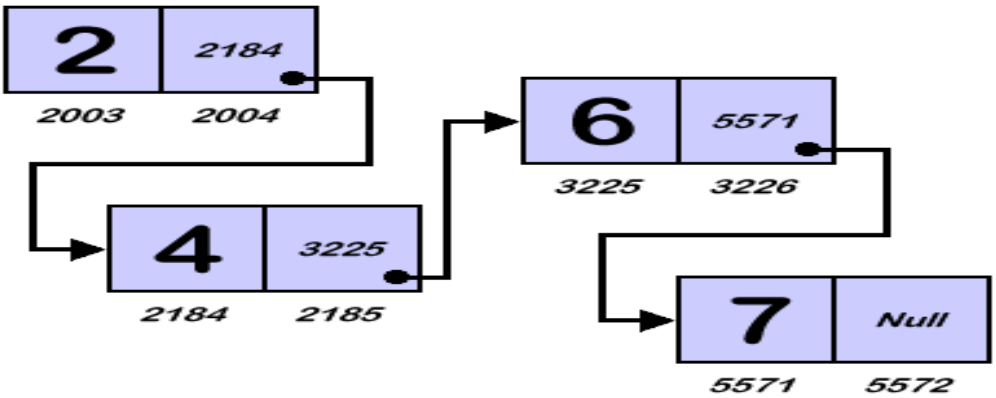
משימה מס' 2 : מימוש מבנה נתונים מחסנית (LIFO) בעזרת רשימה מקושרת (22 נקודות):

Stack.h
Stack.cpp
Tester.cpp (main)

הקובץ Tester.cpp מופיע בהמשך השאלה , אתם נדרשים להגיש אותו ללא שום שינוי.

תיאור העבודה:

רשימה מקושרת, הינה מבנה נתונים אשר מורכב ממספר צמתים, כל צומת מכילה מידע ומצביע לצומת הבאה (בצומת האחרונה נשמר ערך NULL).



מחסנית היא סוג של מבנה נתונים מופשט הפועל בצורה דומה לזו של מחסנית רובה :האיבר שנכנס ראשון למחסנית יוצא ממנה אחרון (תכונה זו מכונה נכנס אחרון יוצא ראשון LIFO).

יש ליצור מחלקה בשם Stack , המייצגת מחסנית בעזרת רשימה מקושרת.
יש ליצור עבור המחלקה :

- בנאי ברירת מחדל (היוצר מחסנית ריקה)
- בנאי מעתיק (יוצר מחסנית חדשה כעותק של מחסנית-פרמטר),
- הורס שמשחרר את המחסנית (את כל הצמתים של המחסנית)
- אופרטור השמה = המעדכן את האובייקט מסוג הנ"ל הנוכחי (LValue) עי" אובייקט הפרמטר (RValue).

יש ליצור את האופרטורים הבאים עבור המחלקה לטיפול במחסנית:

1. אופרטור **+=**, אשר מקבל מספר שלם ומוסיף אותו למחסנית לפי עיקרון האיחסון שעליו המחסנית מתבסס .
2. אופרטור **-=**, אשר מקבל מספר שלם המהווה את מספר הערכים שיש להוציא מהמחסנית ומוציא אותם בהתאם .במידה והמספר גדול מכמות הערכים שהמחסנית מאחסנת יש להציג הודעה מתאימה ולא לנקוט בשום פעולה נוספת.
3. אופרטור **==** הבודק האם 2 המחסניות *זהות* (ז"א הן מכילות את אותם הערכים באותו הסדר בדיוק).
4. אופרטור **!=** הבודק האם 2 המחסניות *שונות* (ז"א אם הן לא באותו גודל/בעלות סדר שונה/לא מכילות את אותם הערכים).
5. אופרטור ! ההופך את ארכי המחסנית , כפי שמופיע בדוגמא מטה.
6. אופרטור פלט **<<** אשר מדפיס למסך את תוכן המחסנית באופן מסודר, כפי שמופיע בדוגמא מטה.

ניתן להוסיף מתודות פרטיות כרצונכם.

אנו נגדיר בתוך מחלקת רשימה (מחסנית) מחלקה פנימית פרטית (private inner class) בשם Node (קודקוד/צומת).
שימו לב כי מחלקה פנימית היא מחלקה שההצהרה שלה נמצאת בתוך מחלקה אחרת – מחלקת האם שלה (במקרה זה מחלקת רשימה/מחסנית). ולכן כל גישה אליה היא רק דרך מחלקת האם.

למחלקת Node יהיו שתי שדות:

- מספר שלם.
- מצביע ל-Node (next).

להלן הגדרה חלקית של מחלקות (אסור להוסיף שדות מעבר למוגדר !) :

```
class Stack {
        class Node{
                int value;
                Node* next;
        public:
                Node(int a):value(a),next(NULL){}
                Node(const Node &);


        };// end of class Node
        Node* top;
        int size;
public:
```

```
}; // end of class Stack

Stack::Node::Node(const Stack::Node & other): value ( other.value) {
// if there's another node after other recursively call copy constructor to create new copy of
node after other
                        if (other.next!=NULL )
                                next=new Node( *(other.next) ) ;
                        else
                                next=NULL;
}
```

דוגמא לשימוש במחלקה Stack (driver) :

```
int main(){
        Stack first, second;
        for (int i = 10; i < 15; i++){
                first += i;
                second += i;
        }
        cout << " First Stack  :" << endl << first <<  " Second Stack  :" <<endl << second <<
endl;
        if (first == second)
                cout << "the stacks are equals !" << endl;
        else
                cout << "the stacks are different !" << endl;
        first -= 1;
        second -= 3;
        cout <<endl<< "After  deleting :" <<endl<< " First Stack  :" << endl << first << "
Second Stack  :" << endl << second << endl;
        if (first != second)
                cout << "the stacks are different !" << endl;
        else
                cout << "the stacks are equals !" << endl;

        cout << " Reverse form of the first stack :" << endl << !first << endl;

        cout << " Delete top from first stack :" << endl << (first-=1 ) << endl;
        Stack third(first);
        first = second;
        cout << endl << "After  assignment :" << endl << " First Stack  :" << endl << first << "
Second Stack  :" << endl << second << endl;
        cout << " Third stack :" <<endl<< third;
        return 0;
}
```

הפלט (מחייב !) :

```
 First Stack  :
14
13
12
11
10
 Second Stack  :
14
13
12
11
10

the stacks are equals !
After  deleting :
 First Stack  :
13
12
11
10
 Second Stack  :
11
10

the stacks are different !
 Reverse form of the first stack :
10
11
12
13

 Delete top from first stack :
11
12
13

After  assignment :
 First Stack  :
11
10
 Second Stack  :
11
10

 Third stack :
11
12
13
```

עליכם לשלב את התהליך המוצג בדרייבר , פלט של ההרצה ומבנה המחלקה החלקי עם הפונקציונליות הנדרשת לטובת המשימה.

עבודה פוריה !!!

המכללה האקדמית להנדסה סמי שמעון
קמפוס באר שבע ביאליק פינת בזל 84100 | קמפוס אשדוד ז'בוטינסקי 84, 77245 | www.sce.ac.il | חייג: *מהפלאפון