מכללת הדסה, החוג למדעי המחשב
תכנות מונחה עצמים ופיתוח משחקים
סמסטר ב', תשפ"ג

תרגיל 1

תאריך אחרון להגשה:

קמפוס הנביאים – יום א', 26/3/2023, ד' ניסן תשפ"ג, בשעה 23:59
קמפוס שטראוס גברים – יום א', 26/3/2023, ד' ניסן תשפ"ג, בשעה 23:59
קמפוס שטראוס נשים – יום ב', 27/3/2023, ה' ניסן תשפ"ג, בשעה 23:59

מטרות התרגיל:

ריענון של העקרונות שנלמדו בסמסטר א' וחזרה עליהם, בפרט תכנון ממשק ותיכון, העמסת אופרטורים, ירושה ופולימורפיזם וכן שימוש בכלים מהספרייה הסטנדרטית, כולל מצביעים חכמים.

תיאור כללי:

בתרגיל זה נממש "מחשבון פונקציות" ניתן לתכנות. המחשבון מחשב את תוצאות הפעולות המבוקשות על מחרוזות (string) לפי קלט מהמשתמש. המשתמש יכול גם ליצור פעולות מורכבות יותר ולהוסיף אותן למחשבון ואחר כך לבקש חישוב של תוצאה של הפעלתן על מחרוזות מסוימות. לשם הפשטות, בתרגיל זה נחזור לעבוד בטרמינל.

פירוט הדרישות:

המחשבון מחזיק רשימת פונקציות שאותן הוא מסוגל לבצע. שלוש הפונקציות שקיימות כבר בהפעלת המחשבון הן:

- **id(x)** – פונקציית הזהות (מקבל מחרוזת ומחזיר אותה כפי שהיא)
- **swapCase(x)** – פונקציה שמחליפה אות קטנה בגדולה ואות גדולה בקטנה
- **reverse(x)** – היפוך המחרוזת.

מהלך ריצת התוכנית:

בכל פעם המחשבון מדפיס למסך את רשימת הפונקציות הנוכחית, מזכיר את הפקודה לקבלת עזרה, ומחכה לקבלת פקודה מהמשתמש. כשהמשתמש מסיים להכניס את הפקודה, המחשבון מבצע אותה ושוב מדפיס את רשימת הפונקציות העדכנית וכן הלאה.

לכל פונקציה ברשימה יש מספר סידורי (המספרים תמיד רציפים) שמודפס לידה בהדפסת רשימת הפונקציות, ובעזרתה המשתמש יכול להתייחס אליה בפקודות שהוא מקליד.

רשימת הפקודות האפשריות:

הערה: הפקודה שהמשתמש מקליד היא תחילת המילה. בסוגריים מוצגת השלמת המילה המלאה כדי לעזור בהבנת המשמעות. גם במסך העזרה נדפיס כך.

eval(uate) num x – מחשבת את הערך של הפונקציה שמספרה num, עם המחרוזת x (מילה אחת, ללא רווחים). ומדפיסה את התוצאה

substr(ing) num1 num2 – מוסיפה לרשימת הפונקציות פונקציה ליצירת תת מחרוזת באופן הבא: חיתוך מחרוזת החל מהמקום num1 באורך של num2 תווים. הפונקציה מצטרפת יחד עם הערכים שהוכנסו. (ראו בדוגמא)

mul(tiply) n num – מוסיפה לרשימת הפונקציות פונקציה שמפעילה את הפונקציה מספר num, ומשרשרת את התוצאה לעצמה n פעמים.

add num1 num2 – מוסיפה לרשימת הפונקציות פונקציה שהיא שרשור של תוצאת הפונקציות שמספרן num1 ו־num2.

comp(osite) num1 num2 – מוסיפה לרשימת הפונקציות פונקציה שהיא הרכבה של הפונקציות שמספרן num1 ו־num2. המשמעות היא שתחילה מפעילים את הפונקציה שמספרה num1 ועל התוצאה מפעילים את הפונקציה שמספרה num2.

del(ete) num – מוחקת את הפונקציה שמספרה num מרשימת הפונקציות. שימו לב שאם יש פונקציות שנסמכות על הפונקציה הזו (פונקציות שיצרנו תוך שימוש בפונקציה הזו כאבן בניין) הפונקציות הללו עדיין צריכות להמשיך לעבוד כרגיל. מצד שני, מספרי הפונקציות שנשארו ברשימה צריכים להישאר רציפים, כמו שהוזכר לעיל.

help – מדפיסה מסך עזרה עם רשימת הפקודות האפשריות והסבר קצר עליהן.

exit – מדפיסה למסך "Goodbye" ויוצאת מהתוכנית.

דוגמה לריצת התוכנית:

הערה: אין צורך לעקוב בצורה מדויקת אחרי הניסוח כאן, אבל כן לעקוב אחרי הקו הכללי.

הקלט מהמשתמש מסומן ברקע צהוב.

הפלט ההתחלתי:

```
List of available string operations:
0.      id
1.      swapCase
2.      reverse
```

פונקציית הזהות

```
Enter command ('help' for the list of available commands): eval 0
HelloWorld
id(HelloWorld) = HelloWorld

List of available string operations:
0.      id
1.      swapCase
2.      reverse
```

היפוך אות קטנה לגדולה וגדולה לקטנה

```
Enter command ('help' for the list of available commands): eval 1
HelloWorld
swapCase(HelloWorld) = hELLOwORLD

List of available string operations:
0.      id
1.      swapCase
2.      reverse
```

פונקציית היפוך סדר אותיות

```
Enter command ('help' for the list of available commands): eval 2
HelloWorld
reverse(HelloWorld) = dlroWolleH


List of available string operations:
0.      id
```

```
1.      swapCase
2.      reverse
```

הוספת פונקציית חיתוך לתת מחרוזת מאינדקס 1 בגודל 5 תווים

```
Enter command ('help' for the list of available commands): substr 1 5

List of available string operations:
0.      id
1.      swapCase
2.      reverse
3.      substr_1_5
```

הפעלת פונקציית החיתוך שהוספנו

```
Enter command ('help' for the list of available commands): eval 3 HelloWorld
substr_1_5(HelloWorld) = elloW

List of available string operations:
0.      id
1.      swapCase
2.      reverse
3.      substr_1_5
```

הוספת פונקציית חיבור של פונקציות

```
Enter command ('help' for the list of available commands): add 2 3

List of available string operations:
0.      id
1.      swapCase
2.      reverse
3.      substr_1_5
4.      reverse + substr_1_5
```

הפעלת פונקציית החיבור שהוספנו

```
Enter command ('help' for the list of available commands): eval 4 HelloWorld
(reverse + substr_1_5)(HelloWorld) = dlroWolleHelloW

List of available string operations:
0.      id
```

```
1.      swapCase
2.      reverse
3.      substr_1_5
4.      reverse + substr_1_5
```

הוספת פונקציית כפל - הפעלה 3 פעמים של פונקציה #2 ושרשור התוצאות

```
Enter command ('help' for the list of available commands): mul 3 2

List of available string operations:
0.      id
1.      swapCase
2.      reverse
3.      substr_1_5
4.      reverse + substr_1_5
5.      reverse * 3
```

הפעלת פונקציית ההכפלה שהוספנו

```
Enter command ('help' for the list of available commands): eval 5
Hello
(reverse * 3)(Hello) = olleHolleHolleH

List of available string operations:
0.      id
1.      swapCase
2.      reverse
3.      substr_1_5
4.      reverse + substr_1_5
5.      reverse * 3
```

הוספת פונקציית הרכבה של פונקציות 5,4

```
Enter command ('help' for the list of available commands): comp 5
4

List of available string operations:
0.      id
1.      swapCase
2.      reverse
3.      substr_1_5
4.      reverse + substr_1_5
5.      reverse * 3
```

```
6.      (reverse * 3)  ->  (reverse + substr_1_5)
```

הפעלת פונקציית ההרכבה שהוספנו

```
Enter command ('help' for the list of available commands): eval 6 HelloWorld
((reverse * 3) -> (reverse + substr 1,5))(HelloWorld) = HelloWorldHelloWorldHelloWorldlroWo

List of available string operations:
0.      id
1.      swapCase
2.      reverse
3.      substr_1_5
4.      reverse + substr_1_5
5.      reverse * 3
6.      (reverse * 3)  ->  (reverse + substr_1_5)
```

מחיקת פקודה

```
Enter command ('help' for the list of available commands): del 4

available string operations:
0.      id
1.      swapCase
2.      reverse
3.      substr_1_5
4.      reverse * 3
5.      (reverse * 3)  ->  (reverse + substr_1_5)

Enter command ('help' for the list of available commands): help
The available commands are:
* eval(uate) num x - compute the result of function #num on the following x string
* substr(ing) num1 num2 - creates a substring starting from index num1 with the length of num2 characters
* add(on) num1 num2 - Creates an operation that is the concatenation of operation #num1 and operation #num2
* mul(tiply) n num - Creates an operation that is the multiply n of operation #num
```

```
* comp(osite) num1 num2 - creates an operation that is the composition of operation #num1 and operation #num2
* del(ete) num - delete operation #num from the operation list
* help - print this command list
* exit - exit the program


List of available string operations:
0.      id
1.      swapCase
2.      reverse
3.      substr_1_5
4.      reverse + substr_1_5
5.      reverse * 3
6.      (reverse * 3)  ->  (reverse + substr_1_5)

Enter command ('help' for the list of available commands): exit
Goodbye!
```

הערות לגבי צורת המימוש:

- לפעולות על המחרוזות אפשר (ורצוי) להשתמש ביכולות של הספרייה הסטנדרטית. הפונקציות הרלוונטיות נמצאות ב-string
- כדאי להשקיע זמן בתיכון התוכנית. עם תיכון נכון, הקוד יהיה די מינימלי ופשוט לכתיבה. בפרט, אל תחששו להשתמש בכלים של הספרייה הסטנדרטית כמו וקטור ומצביעים חכמים, הם יהפכו את הקוד שלכם להרבה יותר נקי, יעיל וקצר ויקלו עליכם בכתיבת קוד שעובד נכון.
- כל המספרים בפונקציות בתוכנית הם int.
- ניתן להניח שהקלט תקין מהבחינה שבמקום שאנחנו מצפים למספר אכן קיבלנו מספר. כן צריך לבדוק את שם הפקודה ואת מספרי הפונקציות.

קובץ ה־README:

יש לכלול קובץ README שיקרא README.doc, README.docx או README.txt (ולא בשם אחר). הקובץ יכול להיכתב בעברית ובלבד שיכיל את הסעיפים הנדרשים.

קובץ זה יכיל לכל הפחות:

1. כותרת.
2. פרטי הסטודנט: שם מלא כפי שהוא מופיע ברשימות המכללה, ת"ז.

3. הסבר כללי של התרגיל.
4. רשימה של הקבצים שיצרנו, עם הסבר קצר (לרוב לא יותר משורה או שתיים) לגבי תפקיד הקובץ.
5. מבני נתונים עיקריים ותפקידיהם.
6. אלגוריתמים הראויים לציון.
7. באגים ידועים.
8. הערות אחרות.

יש לתמצת ככל שניתן אך לא לוותר על אף חלק. אם אין מה להגיד בנושא מסוים יש להשאיר את הכותרת ומתחתיו פסקה ריקה. **נכתוב ב־README כל דבר שרצוי שהבודק ידע כשהוא בודק את התרגיל.**

אופן ההגשה:

הקובץ להגשה: ניתן ליצור בקלות קובץ zip המותאם להגדרות ההגשה המפורטות להלן ישירות מתוך VS, כפי המוסבר תחת הכותרת "יצירת קובץ ZIP להגשה או לגיבוי" בקובץ "הנחיות לשימוש ב־Visual Studio 2022". אנא השתמשו בדרך זו (אחרי שהגדרתם כראוי את שמות הצוות ב־MY_AUTHORS: **נשים את שמות המגישים בתוך המרכאות, נקפיד להפריד בין השם הפרטי ושם המשפחה בעזרת קו תחתי ואם יש יותר ממגיש אחד, נפריד בין השמות השונים בעזרת מקף (מינוס '-')**) וכך תקבלו אוטומטית קובץ zip המותאם להוראות, בלי טעויות שיגררו אחר כך בעיות בבדיקה.

באופן כללי, הדרישה היא ליצור קובץ zip בשם exN-firstname_lastname.zip (או במקרה של הגשה בזוג – exN-firstname1_lastname1-firstname2_lastname2.zip), כשהקובץ כולל את כל קובצי הפרויקט, למעט תיקיות out ו־vs. כל הקבצים יהיו בתוך תיקייה ראשית אחת.

את הקובץ יש להעלות ל־Moodle של הקורס למשימה המתאימה. בכל מקרה, **רק אחד** מהמגישים יגיש את הקובץ ולא שניהם.

הגשה חוזרת: אם מסיבה כלשהי החלטתם להגיש הגשה חוזרת יש לוודא ששם הקובץ זהה לחלוטין לשם הקובץ המקורי. אחרת, אין הבודק אחראי לבדוק את הקובץ האחרון שיוגש.

כל שינוי ממה שמוגדר פה לגבי צורת ההגשה ומבנה ה־README עלול לגרור הורדת נקודות בציון.

מספר הערות:

1. נשים לב לשם הקובץ שאכן יכלול את שמות המגישים.
2. נשים לב לשלוח את תיקיית הפרוייקט כולה, לא רק את קובצי הקוד שהוספנו. תרגיל שלא יכלול את כל הקבצים הנדרשים, לא יתקבל וידרוש הגשה חוזרת (עם כללי האיחור הרגילים).

המלצה כללית: אחרי הכנת הקובץ להגשה, נעתיק אותו לתיקייה חדשה, נחלץ את הקבצים שבתוכו ונבדוק אם ניתן לפתוח את התיקייה הזו ולקמפל את הקוד. הרבה טעויות של שכחת קבצים יכולות להימנע על ידי בדיקה כזו.

בהצלחה!