בס"ד

שפות תכנות – תרגיל 4

תאריך הגשה: 22.01.2023

הוראות הגשה: ההגשה בזוגות. כל זוג נדרש לחשוב, לפתור ולכתוב את התרגיל בעצמו. מותר להתייעץ עם זוגות אחרים אך חל איסור מוחלט להחזיק ולהיעזר בתרגיל כתוב של זוג אחר.
יש לקרוא הוראות אלא בקפידה, הגשה שלא על פי הוראות אלה תוביל להורדת ניקוד ולא יתקבלו על כך ערעורים!

בחלק ב (פרולוג) יש להגיש אך ורק את התשובה ללא תוספות (כמו שורת פקודה להרצת הקובץ שראינו בתרגול).

חומר עזר מומלץ: כדאי להבין היטב את הרצאות ותרגולים מספר 10-12. קישור לתרגולים (המצגות נמצאות בתיאור הסרטון):

https://www.youtube.com/watch?v=4TQIBtY4fRc&list=PLaMkJ2Pfx92I7DbMteYLYmMDyn3N0dDIT&index=10

כמו כן הסברנו על memorization בתרגול על imperative ocaml (תרגול מספר 9 בסופו).

memoization ב-javascript:

https://www.freecodecamp.org/news/understanding-memoize-in-javascript-/51d07d19430e

/https://www.geeksforgeeks.org/javascript-memoization

קישור להסבר על השערת גולבך:

https://davidson.weizmann.ac.il/online/mathcircle/articles/%D7%94%D7%A9%D7%A2%D7%A8%D7%AA-%D7%92%D7%95%D7%9C%D7%93%D7%91%D7%9A-%D7%90%D7%95-%D7%9E%D7%A9%D7%A4%D7%98%D7%99%D7%9D-%D7%A4%D7%A9%D7%95%D7%98%D7%99%D7%9D-%D7%A2%D7%9D-%D7%94%D7%95%D7%9B%D7%97%D7%94-%D7%9C%D7%90-%D7%A4%D7%A9%D7%95%D7%98%D7%94

סקריפט בדיקה: לחלק התכנות בפרולוג מצורף סקריפט בדיקה בשם test.sh שצריך להיות באותה תיקייה עם קטעי הקוד ועם הקובץ test.txt המצורף. כדי להריץ את הסקריפט יש להריץ את הפקודה:

sudo bash test.sh

שימו לב שהסקריפט שולח אזהרה – ניתן להתעלם ממנה. כמו כן הסקריפט בודק את הפלט שלכם עם כל קטעי הקוד ולכן יהיו שגיאות אם לא מימשתם עדיין את כל הסעיפים. שימו לב: ייתכנו מקרים בהם מספר זוגי יכול להיכתב כסכום של שני מספרים ראשוניים. למשל, המספר 24 יכול להכתב כסכום 11+13 אך גם כסכום 19+5. הסקריפט מחזיר רק אופציה אחת כזו וזה בסדר גמור אם התכנית שתממשו תחזיר אופציה אחרת .

ייתכן שתצליחו להריץ את הסקריפט ללא sudo, אם לא ככל הנראה תצטרכו להזין את הסיסמא שלכם בלינוקס.

בנוסף הסקריפט מניח ש-swipl מותקן אצלכם במקום הדיפולטיבי בהינתן שלא צריך לשנות את הכתובת שלו בסקריפט.

בס"ד

אין חובה להריץ את הסקריפט אבל אני אבדוק עם סקריפט דומה ולכן מומלץ.

<u>מה להגיש:</u>

קבצי תכנות מחלק א:

קבצי התכנות של חלק ב:

יש להגיש את כל הקבצים ביחד עם קובץ id.txt בקובץ zip בשם ex4.zip

בס"ד

- **חלק א: javascript**

בתרגיל זה נלמד על Memoization ב-javascript (מומלץ מאוד להסתכל על הקישור בחומר עזר מומלץ שבתחילת התרגיל). בתרגיל יש להגיש כל שאלה בכל בנפרד בשם ex4_<number of quetion>.js. לדוגמא עבור שאלה אחת צריך להגיש את הקובצים e4_1a.js, ex4_1b.js.

1) Memoization is an optimization technique used primarily to speed up computer programs by storing the results of function calls and returning the cached result when the same inputs occur again. Memoization is a specific form of caching that involves caching the return value of a function based on its parameters. Implement Memoization in Javascript for the following:

a) F(n)=n! (F(n)=n*F(n-1), F(0)=1)

b) F(n) returns Fibonacci number n. F(n)=F(n-1)+F(n-2), F(0)=0, F(1)=F(2)=1

2) Implement in JavaScript the high-order function memoize() that accepts a function (name it f for example) as its argument, and returns a memoized version of the function f (in this section you can assume f has only one argument).

3) Based on your memoize() solution function of question 2 (an adaptation might be needed to cope with the recursive call), give another solution to question 1 (1.a and 1.b) put your solution in files ex4_3a and ex4_3b.

4) Generalize memoize() to accept a function f as its argument like in question 2, but now, f may have several arguments (you can assume that arguments of f all have distinct string representations).

- **חלק ב: prolog**

בחלק זה נתכנת בשפת prolog, עבור כל סעיף צריך לכתוב קובץ pl. שהשם שלו הוא שם ה"פונקציה" שאותה יש לכתוב. לדוגמא עבור שאלה אחת צריך להגיש קובץ element_at.pl כך שניתן יהיה להפעיל את הפרדיקט לאחר ביצוע הפקודה .consult(element_at)

1) כתבו פרדיקט בשפת prolog בשם element_at שמקבל משתנה X, רשימה ומספר שלם k כלשהו ומחזיר את האיבר ה-k ברשימה. האיבר הראשון ברשימה הוא איבר מספר 1, ניתן להניח שלא יתקבל k הגדול מאורך הרשימה או קטן שווה לאפס.
דוגמא:

```
?- element_at(X,[a,b,c,d,e],3).
X = c
```

2) כתבו פרדיקט בשפת prolog בשם remove_at שמקבל משתנה X, רשימה ומספר שלם k כלשהו (עד כאן בדיוק כמו התרגיל הקודם) ובנוסף גם משתנה R. הפרדיקט יחזיר במשתנה X את האיבר ה-k (בדיוק כמו השאלה הקודמת) ובמשתנה R הפרדיקט יחזיר את הרשימה ללא האיבר ה-k (שנמצא ב-X).
דוגמא:

```
?- remove_at(X,[a,b,c,d],2,R).
X = b
R = [a,c,d]
```

3) כתבו פרדיקט בשפת prolog בשם is_prime שמקבל מספר שלם כלשהו ומחזיר אמת אם המספר הוא ראשוני ושקר אחרת.
דוגמא:

```
?- is_prime(7).
true

?- is_prime(15).
false
```

4) כתבו פרדיקט בשפת prolog בשם goldbach שמקבל מספר ומשתנה X ומחזיר ב-X רשימה המכילה שני מספרים ראשוניים שסכומם הוא המספר בארגומנט הראשון לפונקציה (במידה ולא קיימים החזירו רשימה ריקה).
דוגמא:

```
?- goldbach(28, X).
X = [5,23]
```

כאשר יש יותר מתשובה אחת ניתן להחזיר רק אחת.

5) הגדירו את הפרדיקטים הבאים בשפת prolog:
and, or, not, xor, nand, nor, equal
כל פרדיקט מקבל שני ארגומנטים (למעט הפרדיקט של not שמקבל ארגומנט אחד) ומחזיר אמת או שקר.
כעת כתבו פרדיקט table המקבל שני משתנים וביטוי בוליאני המשתמש בפרדיקטים שהגדרתם **וכותב למסך** טבלת אמת עבור הביטוי שהתקבל.
דוגמא:

```
?- table(A,B, and(A,(or(A,not(B))))).
true true true
true fail true
fail true fail
fail fail fail
```

ניתן להניח תקינות של הקלט.