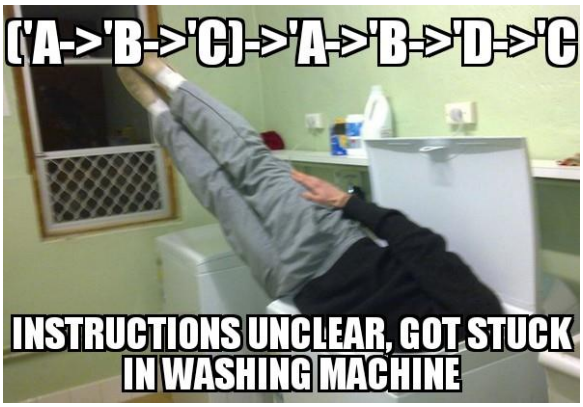
שפות תכנות – תרגיל מספר 2 חלק יבש

תאריך הגשה: *28.11.17*

מגישים:

סהר כהן – 206824088

יובל נהון – 206866832



4. ציינו כמה שגיאות בשקפים של פרק 2.4, אם ישנן כאלה, והציעו תיקון.

שאלה 1: שפה אוטרקית, לדוגמת שפת פסקל, היא שפה ה מושטת לפי הגישה האוטרקית, ולפיה לכל תכנית יש רק קטע קוד **אחד** היוצר אותה, וקטע קוד זה נמצא בקובץ **אחד** בלבד. כלומר, הגישה דוגלת ב"אטימות" מן העולם החיצון במובן המימוש: כל הפונקציות אשר התוכנית יכולה לגשת אליהן צריכות להיות ממומשות אך ורק בתוך קטע הקוד הזה (או לחלופין פונקציות מובנות בשפה, לדוגמת פונקציות קלט/פלט), ולא באף קטע קוד אחר.
כל תכנית הכתובה בשפה אוטרקית מגדירה במפורש את נקודת ההתחלה של התכנית, ואת התחום בקוד הרלוונטי לה. כהמשך לדוגמה שלעיל, בשפת פסקל ישנן המילות השמורות **program** ו – **begin** אשר מגדירות את התחלת התוכנית ואת שורה תחילת הביצוע בהתאמה. מילות שמורות בסגנון זה לא תמצאנה בשפות שאינן אוטרקיות, קרי – הגישה המטאפיזית (לדוגמה, שפת C).

שאלה 2: nameable זוהי תכונה ששייכת לכל דבר שניתן לתת לו שם (בן אדם,מספר,אופניים). Nameable בתרגום לעברית זה בר-שיום.

שאלה 3: לפי הצורה הקנונית (canonical form) – לכל אובייקט יש הצגה ייחודיות המתאימה רק לו. חסרון של שיטה זו גלום בבחירה שרירותית של שיטת אינדוקס (לדוגמת סידור המשתנים בסדר מסוים לפי ערכיי אפשריים). כתוצאה מכך, השוואה בין שני אובייקטים נעשית בעייתית.
לעומת זאת, לפי הגישה הנורמלית (normal form) – רק לאפס יש הצגה ייחודית, ולכל שאר האובייקטים יש הצגה זהה. כך ניתן לחסר בין שני אובייקטים ולהשוות את התוצאה לאפס. אפס ייחודי, ולכן אם התוצאה היא אכן אפס – קיבלנו באופן דטרמיניסטי ששני האובייקטים שווים. אין הבדל בין Backus Normal Form ו – Backus Naur Form.

שאלה 4: בעמוד 7 מתוך 26 במצגת 2.4 ניתנה דוגמה לא תקינה של regular expression:

*(a|b|c|d|e|f) אלו כל המחרוזות שמורכבות אך ורק מהאותיות a,b,c,d,e,f.

בעמוד 9 מתוך 26 במצגת 2.4 בשורה האחרונה כתוב 'e' במקום e.

שאלה 5: לסמל ε מגוון משמעויות שונות ,ובניהן:
א) בהקשר של מתמטיקה, לוגיקה ומדעי המחשב – בתחום השפות הפורמליות משתמשים בסמל זה לצורך סימון ה - string הריק (כלומר, מילה חסרת תווים: "", '', {'\0'}, ""@, {}, ...).
ב) האות החמישית באלפבית היווני, בין האותיות דלתא וזטא.
ג) באסטרונומיה, סמל זה משמש לכינוי שמה של הטבעת החיצונית ביותר של כוכב הלכת אורנוס.
הוכחה שקראנו את המצגת: "epsilon" הוא גם קבוע בשפת MetaPost שמייצג את המספר החיובי הקטן ביותר הניתן לכתיבה בשיטת הייצוג של fixed point. ☺

שאלה 6: ניתן ליצור קבוצות (Sets) של Base Types בלבד.

אפשר לבנות Subrange על בסיס טיפוסים מסוג Integer,Real,Char,על בסיס Subrange גדול ממנו או על בסיס קבוצה.

הערכים ששמורים בתוך מערך יכולים להיות מכל טיפוס אבל המפתחות שלהם חייבים להיות מטיפוסים פרימיטיביים,קבוצה או Subrange.

שאלה 7: לשפת batch של DOS ישנם ה nameables הבאים:

א) משתנים (variables) – ההגבלה על אודות שם תקין עבור משתנה היא זו:
שם של משתנה יכול להכיל אותיות קטנות, אותיות גדולות, מספרים, רווחים ואת

התוויום האלו: @ # $ *) (_ - ' + ~ { } [ ] , . ?
בנוסף, על מנת לקרוא את התוכן של משתנה, יש להקיף את שמו בסימן % בצורה הזו: %<variable>%.

ב) תגיות (labels) – שם של תגית מוגדר בשורה עצמה. השם מתחיל בנקודתיים ומסתיים עם אחד התווים הבאים: רווח " ", נקודתיים ":" או שורה חודשה CR/LF. פקודות בשפה המתייחסות לשם של תגית מוגדרת, לדוגמת GOTO/CALL, תתייחסנה רק ל – 8 התווים הראשונים בשם התגית.

ג) משתנים לקריאה בלבד (read-only variables): ארגומנטים המועברים לתסריט ב batch דרך שורת הפקודה הם משתנים לקריאה בלבד, ובתוך התסריט ניתן לקרוא את התוכן שלהם על ידי שימוש בסימון x% כאשר x הוא ספרה מתאימה לסדר העברת הפרמטרים.

שאלה 8: הטריק בתוכנית הבאה הוא השימוש בstring שמכיל את כל הפקודות להדפסת כל תוכן הקובץ (שימוש ב-# כדי להדפיס char).

```
const a='const a=';b='begin write(a,#39,a,#39#59#98#61#39,b,#39#59#10,b)
end.';

begin write(a,#39,a,#39#59#98#61#39,b,#39#59#10,b) end.
```

שאלה 9:

a) בהנדסת תוכנה, backend & frontend (בעברית: קצה קדמי וקצה אחורי) הם מושגים כלליים אשר מתייחסים לחלקים שונים במערכת התוכנה. ה frontend אחראי על איסוף ועיבוד הקלט אשר הגיע מהמשתמש ומשמש ממשק בין המשתמש וה – backend. תפקיד ה backend הוא לעבד את המידע.

בהקשר של מהדרים: ה frontend מתרגם את הקוד אותו מקבל לייצוג ביניים, וה – backend מעבד את ייצוג הביניים על מנת להפיק את הקלט בשפת המכונה. במהדרים גם קיימות אופטימיזציות בשלב זה אשר פועלות ליצירת קוד שרץ מהר יותר. קיימים מהדרים המאפשרים בחירה בין סוגי front ends ו back ends שונים אשר מאפשרים ביצוע ניתוח תחבירי של קוד בשפות שונות, והפקת שפות מכונה המתאימים לסוגי מעבדים שונים.
ב – kotlin: יש תמיכה ב backend עבור Web, JVM ו -JavaScript.

b) מנגנוני הגדרות השפה שבהן משתמשת שפת Kotlin הן:
דקדוק: קיים מפרט דקדוק בשפה, בעל הגדרות דקדוקי EBNF ודקדוקי BNF עם אפשרויות קיצור (syntactic sugar) לדוגמת אופרטור ++ וכדומה. בדקדוק של השפה, אין חובה על שימוש בנקודה-פסיק ";" בסוף כל פקודה. השפה גם מגדירה באמצעות ביטויים רגולריים את השמות המותרים עבור nameables. כמובן, הצורך ב BNF ובהרחבה EBNF מגיע מהכוח המוגבל של הביטויים הרגולריים על אודות הגדרת syntax השפה כולה. דוגמה: בשפת Kotlin, על מנת לתעד את הקוד, רושמים "**/" לצורך פתיחת קטע שאיננו קוד בשפה ומתאר הערות עבור הקורא בלבד, ו – "/*" על מנת לסגור.
בנאים, איברים אטומיים ואיברים מרוכבים: קיימים בשפה מספר בנאים שונים עבור טיפוסים שונים. איברים אטומים (קרי, פרימיטיביים) ואיברים מרוכבים (קרי, מחלקות) וכדומה.

c) שפת Kotlin אינה שפה אוטרקית כמו pascal, שכן נקודה תחילת ביצוע התוכנית מוגדרת ומוסכמת והיא החל מהשורה fun main (args: Array), ולתוכנית אין גבולות מוגדרים, משום שכל תוכן הקוד בפונקציית main יכול לקרוא לפונקציות המוגדרות בקבצי קוד שונים. מכאן שלתוכנית יכולים להיות מספר קבצי קוד הבונים אותה, בניגוד לעקרון המנחה של הגישה האוטרקית.

d) ה nameables בשפת kotlin הם: משתנים(variables), פונקציות, פקטות(packages), אובייקטים ותוויות (מסתיימות ב - @)

e) מותר לתת אותו השם לישויות שונות בשפת Kotlin, לדוגמה: ערך בשם var1 ותווית @var1 הן ישויות שונות החולקות שם משותף, אבל הדבר אינו פוגע בפיענוח הקוד על ידי הקומפיילר (אין הגבלה על אקסקלוסיביות של שמות, בתנאי שאינם עבור אותה היישות).

f) בנאי הטיפוסים בשפת Kotlin הם:
א) Subrange Types – טיפוסים אשר מכילים ערך בודד אשר חסום על ידי טווח ערכים מוגדר (למשל, כפי שמפורט במפרט של שפת Kotlin, רוחב ביט של טיפוס פרימיטיבי double הוא 64, int הוא 32, וכו'). כל הערכים בטווח הם מאותו הטיפוס.
ב) Array Types – מערכים בגודל קבוע. הערכים במערך מוגדרים מראש להיות טיפוס כלשהו, ואורך המערך נקבע בזמן אתחול (כמספר שלם).
ג) Kotlin – Class Types אנאלוגית מאוד ל – Java, ועל כן, כשפה מונחית עצמים, כל עצם הוא למעשה class. אם כן, קיים גם בנאי למטרת יצירת מחלקות (שקול ל"מערך בגודל בלתי מוגדר"). אובייקטים רבים הם מטיפוס class, לדוגמת enum, file וכדומה.