עקרונות קומפילציה – עבודה 1

הנחיות:

- הגשת העבודה **בזוגות**.
- רק אחד מבני הזוג מגיש את העבודה בתיבות ההגשה.
- יש להגיש כל חלק לתיבת ההגשה הייעודית שלו.
- כתיבה בכתב קריא.
- יש להציג ולהסביר את צעדי החישוב שביצעתם. פתרונות ללא הסבר לא יתקבלו.
- כל שאלה על העבודה יש לשאול בפורום הייעודי לעבודה במודל.
- אין לפרסם פתרונות בפורום הייעודי בעת שאלת השאלה.
- שאלות פרטניות יש לשלוח למתרגל האחראי - אלכסנדר לזרוביץ' במייל alexla@ac.sce.ac.il

הערות הגשה:

- חלק תיאורטי – יש להגיש קובץ PDF בודד.
- חלק מעשי – יש להגיש קבצי lex ו-yacc וקבצי עזר במידה וקיימים.

בהצלחה!

חלק תיאורטי(10 נק') – RegEx, NFA, DFA

בהינתן הביטוי הרגולרי r:

$$\boldsymbol{r = (a+b)(aa+ba+ab)^*(ab+b)}$$

1. בנו NFA מהביטוי הרגולרי r בעזרת אלגוריתם Thompson.
2. בנו וציירו DFA המתקבל מה-NFA בעזרת אלגוריתם Subset Construction.
3. צמצמו DFA אם אפשר, יש להראות את הדרך כפי שלמדתם בהרצאות ובתרגולים.

חלק מעשי(90 נק') – Scanner and Parser

בחלק זה עליכם לכתוב scanner בLEX ו-parser ב-YACC לשפה שתיאורה פורסם במודל, חלק זה יהווה את החלק הראשון בPipeline של הקומפיילר אותו נבנה.
ה-parser אמור לקבל **כל קוד אפשרי** בשפה הנ"ל ולבנות(Abstract Syntax Tree (AST. אם יש טעות תחבירית כלשהי, על ה-parser להדפיס הודעת שגיאה בהתאם לסוג הטעות ויש לפרט מה בדיוק הטעות (ככל האפשר, מספר שורה וסיבת הטעות).
יש לכתוב פונקציה אשר סורקת עץ ב- preorder ומדפיסה את ה-AST במידה והקוד תקין. על הפונקציה להשתמש בהדפסה בסוגריים ובהזחות (בדומה לדוגמא) על מנת להדגיש את הקינון של הקוד.

לדוגמא, עבור הקוד:

```
def foo(int x, y, z; bool f):{
        if f:
                x = x + y;
        else : {
                y = x + y + z;
                z = y * 2;
        }
}


def goo()->string:{
        return 'a';
}
```

הדפסה של העץ:

הערה: ההדפסה לא צריכה להיות בדיוק אותו דבר. זאת רק דוגמה לפלט אפשרי.

```
(CODE
	(FUNC
		foo
		(ARGS
			(INT
				x
				y
				z
			)
			(BOOL
				f
			)
		)
		(RETURN VOID)
		(BODY
			(IF-ELSE
				(f)
				(BLOCK
					(ASS x
						(+
							x
							y
						)
					)
				)
				(BLOCK
					(ASS y
						(+
							(+
								x
								y
							)
							z
						)
					)
					(ASS z
						(*
							y
							2
						)
					)
				)
			)
		)
	)
	(FUNC
		goo
		(ARGS NONE)
		(RET STR)
		(BODY
			(RET ‘a’)
		)
	)
)
```