שפות תכנות - תרגיל בית 5:

מגישים:

אפק נחום 214392706

עידו טאוסי 214008997

שאלה 1:

1. בתור חוקרים בפקולטה למדעי המחשב, המצאתם שפה חדשה, ורציתם להוסיף טיפוס שנקרא *OneValue* היכול להכיל ערך אחד בלבד. למזלכם, למדתם את הקורס "שפות תכנות", מהו הבנאי בשפת *Mock* המתאים לטיפוס זה?

א. *Bottom* (*None*).

ב. *Unit*.

ג. *Top*.

ד. אין בנאי לטיפוס זה בשפת *Mock*.

תשובה: ב'. למדנו בהרצאה כי רק טיפוס *Unit* מכיל ערך אחד בלבד. סטודנטים יכולים לחשוב שהתשובה תהיה א', אך טיפוס *Bottom* לא מכיל ערכים כלל.

2. במהלך יום הלימודים, נתקלתם בד"ר חיים מהפקולטה לביולוגיה, שרצה להודות לכם על העזרה במהלך הסמסטר, ויש לו בקשה

אחרונה: ד"ר חיים למד לתכנת ועכשיו הוא יודע להכין את העבודות שלו בעצמו, ללא עזרה של סטודנטים. האם תוכלו לעזור לו לבחור דרישות לשפת תכנות לפרויקט? השפה צריכה לענות על הדרישות הבאות:

- השפה צריכה לבדוק את הטיפוסים בזמן קומפילציה בלבד, מפני שלד"ר חיים חשוב זמן ריצה מהיר ביותר.
- בנוסף, ד"ר חיים למד תכנות בשפת *Java* ולכן רוצה לעבוד עם מערכת טיפוסים הפועלת באותו סגנון.

א. *strongly typed, dynamic typing*

ב. *weakly typed, dynamic typing*

ג. *strongly typed, static typing*

ד. *weakly typed, static typing*

תשובה: ג'. המשמעות של בדיקת טיפוסים בזמן קומפילציה היא שפה המקיימת *static typing*. בנוסף, לשפת *Java* היא שפה בעלת מערכת טיפוסים חזקה ולכן נדרשת שפה *strongly typed*.

שאלה 2:

1. a. מבין שלושת האופרטורים הנתונים, האופרטור הלא סימטרי הוא is. אופרטור זה משערך רק את הביטוי הימני, למשל נראה בדוגמה הבאה כי $3\ is\ 1+2$ יחזיר אמת כי $1+2$ ישתערך ל-3, שאכן שווה ל-3, והביטוי $1+2\ is\ 3$ יחזיר שקר, כי 3 אינו שווה לביטוי $1+2$.

```
?- 3 is 1+2.
true.

?- 1+2 is 3.
false.
```

b. נרצה להשוות בין השערוך של הביטויים: $1+3$, $2+2$, שעבור שניהם השערוך הוא 4 ולכן יחזיר אמת כאשר נשתמש באופרטור =:= שמשערך את שני האגפים, ושקר כאשר נשתמש באופרטור = מפני שהם לא מתאימים.

```
?- 1+3 =:= 2+2.
true.

?- 1+3 = 2+2.
false.
```

c. נרצה לעשות אתחול ל-X עם הערך 1. נשים לב שהאופרטור = מסוגל לבצע אתחול למשתנים שלא אותחלו עדיין, אך האופרטור =:= אינו מסוגל ולכן תתרחש שגיאה.

```
?- X =:= 1.
ERROR: Arguments are not sufficiently instantiated
ERROR: In:
ERROR:   [10] _19418=:=1
ERROR:    [9] toplevel_call(user:user: ...) at /usr/lib/swipl/boot/toplevel.pl:1158
?- X = 1.
X = 1.
```

2. ניתן להחליף את אופרטור =:= ב- = # כאשר שני האגפים ניתנים לשיערוך (ואם קיימים בהם משתנים הם צריכים להיות

מאותחלים).

ניתן להחליף את אופרטור is ב- $=$ # כאשר האגף הימני ניתן לשיערוך, והאגף השמאלי הוא אטומי או משתנה שלא אותחל, מפני שאחרת אם האגף השמאלי הוא לא אטומי, הוא ישוערך ב- $=$ # אך לא ב-is.

ניתן להחליף את $=$ ב- $=$ # כאשר שני האגפים הם ביטויים אטומיים, מפני שבכל מקרה אחר האופרטור $=$ # ינסה לשערך את הביטויים, בזמן שאופרטור $=$ לא ישערך.

שאלה 3:

1. תאימות בין טיפוסים בשפת *Typescript* היא *structural*, נדגים זאת בעזרת הדוגמה הבאה, שבה הגדרנו שני טיפוסים *Record*1, *Record*2 אשר שווים מבחינת מבנה, ופונקציה המקבלת *Record*1 ומדפיסה את השדות של האוביקט, והצלחנו להעביר לה ארגומנט מטיפוס *Record*2 ולא התרחשה שגיאה. אם התאימות הייתה *nominal* הייתה מתקבלת שגיאה כיוון ש- *Record*1, *Record*2 הם טיפוסים שונים.

```
type Record1 = { x: number, y: number };
let r1 : Record1 = {x:1, y:2};

type Record2 = { x: number, y: number };
let r2 : Record2 = {x:1, y:2};


function printRecord1(record: Record1): void {
  console.log(`x: ${record.x}, y: ${record.y}`);
}

printRecord1(r1);
printRecord1(r2);
```

והפלט לפי האתר:

```
[LOG]: "x: 1, y: 2"
[LOG]: "x: 1, y: 2"
```

בנוסף, גם תאימות של טיפוסים המוגדרים ע"י מחלקות היא מסוג *structural*, נדגים זאת בעזרת הדוגמה הבאה, שבה הגדרנו שני טיפוסים *Person*1, *Person*2 אשר שווים מבחינת מבנה, ופונקציה המקבלת *Person*1 ומדפיסה את השדות של האוביקט, והצלחנו להעביר לה ארגומנט מטיפוס *Person*2 ולא התרחשה שגיאה. אם התאימות הייתה *nominal* הייתה מתקבלת שגיאה כיוון ש- *Person*1, *Person*2 הם טיפוסים שונים.

```
class Person1 {
  name: string;
  age: number;

  constructor(name: string, age: number) {
    this.name = name;
    this.age = age;
  }

}

class Person2 {
  name: string;
  age: number;

  constructor(name: string, age: number) {
    this.name = name;
    this.age = age;
  }

}

function printPerson1(person: Person1): void {
  console.log(`name: ${person.name}, age: ${person.age}`);
}

let person1 = new Person1("Obi-Wan", 30);
let person2 = new Person2("Anakin", 20);

printPerson1(person1);
printPerson1(person2);
```

והפלט לפי האתר:

```
[LOG]: "name: Obi-Wan, age: 30"
[LOG]: "name: Anakin, age: 20"
```

2. דוגמת קוד בה לכל הערכים יש טיפוס סטטי מוגדר, אך בזמן ריצה עדיין תתרחש שגיאת טיפוס:

```
class A {
  constructor() {}
}

class B extends A {
  my_B: number;

  constructor(my_B: number) {
    super();
    this.my_B = my_B;
  }

  methodB(): void {
    console.log(`Method B in class B. Property B: ${this.my_B}`);
  }
}

class C extends A {
  my_C: boolean;
  my_C2: number;

  constructor(my_C: boolean, my_c2: number) {
    super();
    this.my_C = my_C;
    this.my_C2= my_c2;
  }
}


function crashSystem(arr : A[]) : void {
    arr[0] = new C(true, 30);
}

const arrayB: B[] = [
  new B(10),
  new B(20),
  new B(30)
];

crashSystem(arrayB);
arrayB[0].methodB();
```