שפות תכנות – מטלה 1

אלעזר פיין

1.

✓ x,y

✓ (x)

✗ (z](x]

✗ (x),(y)

✓ [y]

✓ ([x, y])

✗ ((([X])))

✓ [x , y, z]

✓ ([x , y, z])

✓ ((([x , y, z, x])))

2.

<palindrome> ::= <char> | A<palindrome>A | B<palindrome>B | C<palindrome>C | D<palindrome>D
| E<palindrome>E | ... | x<palindrome>x | y<palindrome>y | z<palindrome>z

<char> ::= "" | "A" | "B" | "C" | "D" | "E" | "F" | "G" | "H" | "I" | "J" | "K" | "L" | "M" | "N" | "O" | "P" | "Q" | "R" | "S" | "T" | "U" | "V" | "W" | "X" | "Y" | "Z" | "a" | "b" | "c" | "d" | "e" | "f" | "g" | "h" | "i" | "j" | "k" | "l" | "m" | "n" | "o" | "p" | "q" | "r" | "s" | "t" | "u" | "v" | "w" | "x" | "y" | "z"

3.

```
/*
 *          sign | operation | return
 *          ------------------------
 *            0  |     0     |   b
 *            0  |     1     |   a
 *            1  |     0     |   a
 *            1  |     1     |   b
 */
int getMinMax(int a, int b, char symbol) {
    int operation = (symbol & 2) >> 1;      // 1: get max, 0: get min
    int sign = !(1 + ((a - b) >> 31));      // 1: a < b, 0: a > b
    int xor = sign ^ operation;
    return xor * a | !xor * b;
}
```

4.

כל הSCOPE של הלולאה החיצונית (שמקדמת את i).

כאשר עושים " break outer; " אז יוצאים מה SCOPE הזה. פה לדוגמא ירוץ כך: i = 0 -> j = 0 -> j = 1 -> break;.

5.

.א (Insertion Sort, can be used with any size)

```
void sortOnlyFor(int *arr, int size) {
    for (int i = 0; i < size; i++) {
        int key = arr[i];
        int j = i - 1;
        for (; j >= 0 && arr[j] > key; j--) {
            arr[j + 1] = arr[j];
        }
        arr[j + 1] = key;
    }
}
```

.ב (Insertion Sort, can be used with any size)

```
void sortOnlyWhile(int *arr, int size) {
    int i = 0;
    while (i < size) {
        int key = arr[i];
        int j = i - 1;
        while (j >= 0 && arr[j] > key) {
            arr[j + 1] = arr[j];
            j--;
        }
        arr[j + 1] = key;
        i++;
    }
}
```

.λ

```
void sortOnlyIf(int *arr) {
    if (arr[1] < arr[0])
        swap(&arr[1], &arr[0]);

    if (arr[2] < arr[1]) {
        swap(&arr[2], &arr[1]);
        if (arr[1] < arr[0])
            swap(&arr[1], &arr[0]);
    }

    if (arr[3] < arr[2]) {
        swap(&arr[3], &arr[2]);
        if (arr[1] < arr[0])
            swap(&arr[1], &arr[0]);
        if (arr[2] < arr[1]) {
            swap(&arr[2], &arr[1]);
            if (arr[1] < arr[0])
                swap(&arr[1], &arr[0]);
        }
    }

    if (arr[4] < arr[3]) {
        swap(&arr[4], &arr[3]);
        if (arr[3] < arr[2]) {
            swap(&arr[3], &arr[2]);
            if (arr[1] < arr[0])
                swap(&arr[1], &arr[0]);
            if (arr[2] < arr[1]) {
                swap(&arr[2], &arr[1]);
                if (arr[1] < arr[0])
                    swap(&arr[1], &arr[0]);
            }
        }

    }
}
```

.ד (Insertion Sort, can be used with any size)

```
void sortWithGoTo(int *arr, int size) {
    int i = 0;
    outer_loop:
    if (i < size) {
        int key = arr[i];
        int j = i - 1;

        inner_loop:
        if (j >= 0 && arr[j] > key) {
            arr[j + 1] = arr[j];
            j--;
            goto inner_loop;
        }
        arr[j + 1] = key;
        i++;
        goto outer_loop;
    }
}
```

6.

```
import math


class GeometricShape:
    def __init__(self, length: float):
        self.length: float = length

    def print_area(self):
        raise NotImplementedError()  # act as abstract method

    def print_circumference(self):
        raise NotImplementedError()  # act as abstract method


class Circle(GeometricShape):
    def print_area(self):
        print(f"Circle Area: {math.pi * self.length * self.length}")

    def print_circumference(self):
        print(f"Circle Circumference: {2 * math.pi * self.length}")
```

```python
class Rectangle(GeometricShape):
    def __init__(self, side_a: float, side_b: float):
        super().__init__(side_a)
        self.side_b = side_b

    def print_area(self):
        print(f"Rectangle Area: {self.length * self.side_b}")

    def print_circumference(self):
        print(f"Rectangle Circumference: {2 * (self.length + self.side_b)}")


class Square(Rectangle):
    def __init__(self, length: float):
        super().__init__(length, length)


class Triangle(GeometricShape):
    def __init__(self, side_a: float, side_b: float, side_c: float):
        super().__init__(side_a)
        self.side_b = side_b
        self.side_c = side_c

    def print_area(self):
        s = (self.length + self.side_b + self.side_c) / 2
        area = math.sqrt(s * (s - self.length) * (s - self.side_b) *
                                                  (s - self.side_c))
        print(f"Triangle Area: {area}")

    def print_circumference(self):
        print(f"Triangle Circumference: {self.length + self.side_b +
                                                      self.side_c}")


if __name__ == '__main__':
    indicators = {
        Circle: "C",
        Rectangle: "R",
        Square: "S",
        Triangle: "T",
    }
    shape_ind = None
    while shape_ind not in indicators.values():
        print("Select Shape:")
        for shape in indicators.keys():
            print(f"{indicators[shape]}) {shape.__name__}")
        shape_ind = input("").upper()

    shape = None
    while shape is None:
        try:
            if shape_ind == indicators[Circle]:
                shape = Circle(float(input("Please Enter The Circle's
                                                        Radius: ")))
            elif shape_ind == indicators[Rectangle]:
                shape = Rectangle(*map(float, input("Please Enter The
```

```
                                Rectangle's 2 bases as 'a b' : ").split()))
            elif shape_ind == indicators[Square]:
                shape = Square(float(input("Please Enter The Square Side
                                                                Length: ")))
            elif shape_ind == indicators[Triangle]:
                shape = Triangle(*map(float, input("Please Enter The
                                Triangle's 3 sides as 'a b c' : ").split()))
            else:
                raise ValueError
        except ValueError:
            print("Invalid Input! Try Again...")

    shape.print_area()
    shape.print_circumference()
```

7.

```
# 1
a = 1


class A:
    global a
    print("class A : ", a)
    a = 2

    def __init__(self):
        self.a = 2
        print("class A:: init ", a)

    def thefunc(self):
        a = 3
        print("class A:: thefunc ", a)


class B(A):
    A.a = 4

    def __init__(self):
        self.a = 28


class C(B):
# 2
#######################
    print(a)

    def __init__(self):
        j = super()
        print(j.a)
# 4
        p = A()
        print(p.a)
```

```
    def thefunc(self):
        print(B.a)


class D(C, B):
    a = 4

    def __init__(self):
        global a
# 3
######################
# 5
a = 7

a1 = A()
b = B()
c = C()
d = D()
a1.thefunc()
b.thefunc()
d.thefunc()
```

לא השלמתי ב2 ו3 כי לא היה צורך ורק היה פוגע...

8.

א. הבדל 1: בפייתון ניתן לרשת מכמה מחלקות באותו זמנית, כאשר לא בהכרח קיים ביניהם קשר, בג'אבה אפשר לרשת מכמה מחלקות רק בשרשור (כל אחת יורשת מן השנייה בסדר), או לממש ממשקים בעלי מתודות אבסטרקטיות או מתודות ממומשות דפולטיבית.
הבדל 2: בג'אבה קיימים Access Modifiers (public, protected, package-private and private) אשר מאפשרים encapsulation לאובייקטים שלנו כאשר הדפולטיבי הוא package-private, בפייתון הם קיימים אך בצורה אחרת: שמים underscore ("_") לפני המשתנה על מנת להכריז על רמת הגישה (ללא – public, אחד – protected, שניים – private).
הבדל 3: בג'אבה קיימים primitive types (int, char, bool, float, etc...) ואילו בפייתון כל הסוגים הם מסוג אובייקט.
הבדל 4: בפייתון ניתן להוסיף דינמית לאובייקט שכבר קיים\הוכרז משתנים חדשים, בניגוד לג'אבה.

ב. 1: מימוש של פייתון נותן יותר חופש פעולה למתכנת אך יוצר צורך בהתחשבות נוספת במקרים בהם יש התנגשות בין המחלקות המורישות, מימוש של ג'אבה מגביל יותר אך מבטיח התנהגות שקל לצפות מראש, בנוסף ניתן לפתור הרבה בעיות בעזרת ממשקים.
2: מימוש של ג'אבה קריא וברור יותר ובנוסף יותר נוקשה לעומת פייתון.
3: מימוש בעזרת אובייקטים נותן יותר כלים לעבודה עם הטייפים, (מתודות, הורשה...) ג'אבה ראו צורך זה גם, והוסיפו מחלקות בנוסף לפרימיטיבים (Character, Boolean, Integer,etc...).
4: המימוש הפייתון נותן הרבה (אולי יותר מדי) חופש פעולה להתעסק עם האובייקטים, בניגוד לג'אבה בה יש יותר הגבלות אבל הרבה יותר בטוח וקונקרטי.

ג. מימוש מחלקה היורשת 2 מחלקות.
לדוגמא, מחלקות: חייל (נושא נשק: BOOL), חולה (ימי מחלה: INT), וחייל חולה היורש מחייל וגם מחולה.

9.

```java
import java.util.ArrayList;
import java.util.List;

public class SistersPizza extends PlainPizza implements OlivesMixin {
    public void prepare_pizza() {
        add_olives(this);
    }
}

class PlainPizza {
    List<String> toppings;

    PlainPizza() {
        toppings = new ArrayList<>();
    }
}

interface OlivesMixin {
    default void add_olives(PlainPizza pizza) {
        System.out.println("Adding Olives");
        pizza.toppings.add("olives");
    }
}
```

10.

```c
#include <stdlib.h>
#include <stdio.h>

typedef struct {
    char **toppings;
    int toppings_size;
} PlainPizza;

void initPlainPizza(PlainPizza *pizza) {
    pizza->toppings_size = 0;
    pizza->toppings = NULL;
}

void add_olives(PlainPizza *pizza) {
    puts("Adding Olives!");
    pizza->toppings = realloc(pizza->toppings, (pizza->toppings_size + 1) *
                                                    sizeof(char *));
    pizza->toppings[pizza->toppings_size++] = "olives";
}

typedef struct {
    void (*add_olives)(PlainPizza *pizza);
} OlivesMixin;
```

```c
void initOlivesMixin(OlivesMixin *olivesMixin) {
    olivesMixin->add_olives = add_olives;
}

typedef struct {
    PlainPizza plainPizza;
    OlivesMixin olivesMixin;
    void (*prepare_pizza)(PlainPizza *pizza);
} SistersPizza;

void initSistersPizza(SistersPizza *sistersPizza) {
    initPlainPizza(&sistersPizza->plainPizza);
    initOlivesMixin(&sistersPizza->olivesMixin);
    sistersPizza->prepare_pizza = sistersPizza->olivesMixin.add_olives;
}

int main() {
    SistersPizza sistersPizza;
    initSistersPizza(&sistersPizza);
    sistersPizza.prepare_pizza(&sistersPizza.plainPizza);

    free(sistersPizza.plainPizza.toppings);
    return 0;
}
```