Introduction to Computer Science Application

Exercises

Ugo Dal Lago* Marco Di Felice†

January 21, 2010

1 Variables, Expressions and Statements

Exercise 1.
Write three programs that print the values `12`, `"Bioinformatics in Bologna"` and `3.67`, respectively.

Exercise 2.
Given the following sequence of characters, decide which of them are valid variable names.

```
tortellini
$tortellini
tortellini_in_brodo
123tortellini
t123ortellini
```

Exercise 3.
Draw state diagrams along the execution of the following program:

```
a=34
b="Tortellini"
print b
a=14
print a
c=7.5
```

Exercise 4.
How is the state diagram after the execution of the following program:

```
a=34+1
b="Tortellini"
print b+" in brodo"
a=a-16
print a
```

What is the output?

Exercise 5.
What is the output of the following program? Use state diagrams if useful.

```
a=3
b=a+a**2-1
c=b/5
a=c*3-1
print a
print b
print c
```

Exercise 6.
What is the output of the following program?

```
a=3
a=a*a-3*(a+(a-2)*2)
print a
```

Exercise 7.
Write a program that prints the natural numbers between 1 and 5, both included, one per line. Modify it in order to print the numbers consecutively in the same line.

Exercise 8.
A task can be completed in 12176 seconds. Express this quantity in hours, minutes and seconds. To do that, exploits Python capabilities and do not do any calculation by hand.

2 Functions

Exercise 1.
What is the output of the following program?

```
a=4.912
b=78
c="657.16"
print float(c), int(a), float(b), int(float(c)), str(a)
```

Exercise 2.
How is the state diagram after the execution of the following program:

```
a=13
b=a/2
c=a/2.0
```

*dallago@cs.unibo.it
†difelice@cs.unibo.it

```
d=b*1.5
e=d+1
```

Exercise 3.
Compute the length of the diagonal of a rectangle whose vertices has length 7 and 9 centimeters, respectively. To do that, exploits Python capabilities and do not do any calculation by hand.

Exercise 4.
Verify the following equation for 10 different values of x of your choice:

$$\sin^2 x + \cos^2 x = 1$$

To do that, exploits Python capabilities and do not do any calculation by hand.

Exercise 5.
Write a function that prints the natural numbers included between 1 and 5. Moreover, call it twice.

Exercise 6.
What is the output produced by the following program?

```
def a():
  print 1
  b()
  c()
  b()
  print 1
def b():
  print 2
  c()
def c():
  print 3
a()
c()
```

Exercise 7.
What is the output produced by the following program? Use stack diagrams if appropriate.

```
def b():
  a=1
  c()
  a=2*a-3
def c():
  a=3
a=1
b()
print a
```

Exercise 8.
The following program is uncorrect. Why?

```
a=1
def b():
  a=a+1
b()
```

Exercise 9.
Write a function that, given a string as a parameter, prints it in the middle ot two lines fed with the character `-`.

3 Conditionals and Recursion

Exercise 1.
What is the output of the following program?

```
a=10
b=5
c=a%b
d=a/b
print not((d and 1) or (c and 0))
```

Exercise 2.
Write a function `isDivisible(x,y)` which receives in input two parameters `x` and `y` and prints the message `"OK"` if `x` is a multiple of `y`, prints `"NO"` otherwise.

Exercise 3.
Write a function `sqrt(n)` which prints the square root of the parameter `n` if `n>0`. If `n<0`, the function should terminate its execution.

Exercise 4.
Given the `countdown(n)` function, defined as follows:

```
def countdown(n):
    if (n==0):
        print "1"
    elif (n==1):
        return
    else:
        print 2*n
        countdown(n-1)
```

Write the state diagram for `countdown`, called with `n=5`. What is the output of the program?

Exercise 5.
Given the `fact(n)` function, defined as follow:

```
def fact(n):
    if (n==0):
        return 1
    else:
        return n*fact(n-1)
```

write a program which estimates the value of the mathematical costant `e`, by using the following function:

$$e = 1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + ... \quad (1)$$

Your program can stop after summing 5 terms.

Exercise 6.
Modify the program in Exercise 3, so that the parameter `n` is read in input from the keyboard.

Exercise 7.
Given the function `fun(x,y)`, defined as follows:

```
def fun(x,y):
    if 0<x<20:
        if x<y<(x**2):
            print "1"
        else:
            print "0"
    else:
        print "2"
```

Write the output of `fun(x,y)` in the following cases:
(i) `x=10, y=50`
(ii) `x=10, y=101`
(iii) `x=30, y=101`

Exercise 8.
Write a program which reads a string `msg` and a value `n` from keyboard and prints the message `msg` on the display a number of times equal to `n`. Write the program by defining a recursive function `rec(msg,n)`.

4 Fruitful Functions

Exercise 1.
Write the function `farenheit(c)` which returns the Farenheit equivalent to a Celsius temperature `c`:

$$F = \frac{9}{5}c + 32 \quad (2)$$

Exercise 2.
Write a Python function computing the mathematical function defined as follows:

$$\begin{aligned} f(0) &= 3 \\ f(1) &= 5 \\ f(2) &= 7 \\ \forall n \geq 4. f(n) &= f(n-1) \cdot f(n-2) + f(n-3) \end{aligned}$$

Exercise 3.
Using the function `distance(x)` defined at page 49 of the textbook, write the function `equilateral(x1,y1,x2,y2,x3,y3)` which receives as input the coordinates of three points $A = (x_1, y_1)$, $B = (x_2, y_2)$ and $C = (x_3, y_3)$, and returns `True` if ABC is an equilateral triangle, `False` otherwise.

Exercise 4.
Given the following code:

```
def strange(a):
    if isinstance(a,int):
        if a>5:
            return a+ strange(a-1)
        else:
            return strange("NoSense")
    else:
        return 2
```

What is the output of `strange(5)`? What is the output of `strange(7)`? Use state diagrams if useful.

Exercise 5.
Using the `factorial` function (defined at page 54 of the textbook), write the function `euler(n)`, which computes the n-th approximation of the Euler's number:

$$e = 1 + \frac{1}{1!} + \frac{1}{2!} + ... + \frac{1}{n!} \quad (3)$$

Exercise 6.
The function `power(x,n)` defined as follows:

```
def power(x,n):
    if n==0:
        return 1
    elif n==1:
        return x
    else:
        return x*power(x,n+1)
```

should return the value x^n `(n>0)`, but it contains an error. Find the error and fix it.

5 Iteration

Exercise 1.
What is the output of the following program? Use state diagrams if useful.

```
def fun():
    a=30
    return 5
a=20
b=a
b=fun()
print a
print b
```

Exercise 2.
Write a function `gcd(a,b)` which returns the greatest common divisor of two non-zero integers `a` and `b`.

Exercise 3.
What is the output of the following program? Use state diagrams if useful.

```
count=0
i=5
while i>0:
    j=0
    while j<i:
        count=count+(i-j)
        j=j+1
    i=i-1
print count
```

Exercise 4.
An integer number is said to be a perfect number if the sum of its factors (but not the number itself) is equal to the number. For example, 6 is a perfect number, because 6=1+2+3. Write a function `perfect(n)`, which returns `True` whether parameter `n` is a perfect number.

Exercise 5.
Encapsulate the function `perfect(n)` in a function `tablePerfect(m)`, which prints out all the perfect numbers between 1 and `m`.

Exercise 6.
Write a function `prime(n)` which returns `True` iff parameter `n` is a prime number.

Exercise 7.
Encapsulate the function `prime(n)` in a function `tablePrime(m)`, which prints out all the prime numbers between 1 and `m`.

Exercise 8.
Write a program which iteratively reads a value `n` from keyboard and computes $\sum_{i=0}^{n} i$. The program ends when the user types a negative value.

6 Strings

Exercise 1.
Given a string `str`:

```
str="apple"
```

Decide which of the following stamements produce runtime or syntax errors.

- `str2=str[0]`
- `str2=str[-1]`
- `str2=str[len(str)]`
- `str2=str[:]`
- `str[0]="b"`

Exercise 2.
Write a function `replace(str,c,c2)` which receives in input a string `str`, and two characters `c` and `c2` and prints out a string `str2`, where all the occurrence of character `c` are replaced with character `c2`.

Exercise 3.
Write a function `reverse(str)` which prints string `str` in reverse order. Write the function by using a `while` loop. Solve also the exercise by using recursion.

Exercise 4.
Write a program which prints all the possible strings with length 3 which can be created using lowercase letters.

Solution.
A possible implementation is shown in the following:

```
import string
counter=0
for c1 in string.lowercase:
    str=c1
    for c2 in string.lowercase
        str2=str+c2
        for c3 in string.lowercase:
            str3=str2+c3
            print counter," ",str3
            counter=counter+1
```

Exercise 5.

In most computer applications, users are requested to provide robust passwords to perform some specific tasks. A password is said valid if: (i) it is longer than 6 characters and (ii) it contains at least 2 lowercase letters, 3 uppercase letters and 1 digit. Write a function `validPassword(str)` which returns `True` iff `str` is a valid password.

Exercise 6.
A palindrome is a number or text phrase which can be read in the same way backwards or forwards. For example, the following five-digit integers are palindrome: 12321, 55555, 55755. Write a program that reads a number or text phrase and determine whether it is palindrome.

7 Lists

Exercise 1.
Given a list `str`:

```
a=[[1,2],3,4,5,6]
```

Decide which of the following stamements are valid list operations.

- `a[0][0]=5`
- `print a[0][-3]`
- `print a[0][-1]`
- `a[5]='b'`
- `a[4:4]=[2]`

Exercise 2.
In mathematics, the scalar product of two vectors $a = [a_1, a_2, ..., a_n]$ and $b = [b_1, b_2, ..., b_n]$ is defined as:

$$a \cdot b = \sum_{i=1}^{n} a_i \cdot b_i = a_1 \cdot b_i + ... + a_n \cdot b_n$$

Write a function `scalar(a,b)` which returns the scalar product of two vectors `a` and `b` in input. Assume each vector is represented as a list of floating point. The function should print an error message if vectors `a` and `b` have different dimensions (the scalar product is not defined in such a case).

Exercise 3.
What is the output of the following program?

```
a=['a','b',['b','c'],1,2,3]
del a[0]
a[1][0]='a'
c=a[2:4]
d=a[1]
e=c+d
print e
```

Exercise 4.
In mathematics, the multiplication of two matrix $A \in \mathbb{R}^{m \times n}$ and $B \in \mathbb{R}^{n \times p}$ is defined as a matrix $C \in \mathbb{R}^{m \times p}$ where the elements of C are given by:

$$C[i,j] = \sum_{r=1}^{n} A[i,r] \cdot B[r,j]$$

The multiplication is defined between two matrices only if the number of columns of the first matrix (i.e. n) is the same of the number of rows of the second matrix. Using the function `scalar(a,b)`, write a function `multiplication(a,b,c)` which computes the multiplication between matrices `a` and `b` and stores the result in `c`. Assume all the matrices `a,b,c` have dimension $n \times n$. Each matrices is implemented as a list of nested elements, e.g.:

```
[[1,2,3],[3,4,5],[4,6,8]]
```

Exercise 5.
What is the output of the following program?

```
def fun(a):
    return a[2:]
a=[1,2,3,4,5]
b=a
b[3]=6
c=fun(a)
c[2]=3
print c
```

Exercise 6.
Write a program which reads 5 integer numbers from keyboard. As each number is read, it is inserted in a list if and only if it is not a duplicate of a number already read. If the number is a duplicate, the program continue reading from input till a not-duplicated value is digited. When 5 non-duplicated numbers are read, the program computes the average value.

Exercise 7.
Write a function `deleteAll(l,x)` which removes all the occurrence of value `x` from list `l`.

Exercise 8.
Write a program which reads from keyword a string giving the score of a soccer game. The program should be able to recognize all the strings structured in this way:

`Team1 Team2 Score-Score`

where:

- `Team1` $\in$ {`Juventus`, `Milan`, `Inter`, `Roma`, `Lazio`}
- `Team2` $\in$ {`Juventus`, `Milan`, `Inter`, `Roma`, `Lazio`}
- `Team1` $\neq$ `Team2`
- `Score` is a digit.

The program should print "OK" if the string in input is well-structured according to the grammar rules defined above. Otherwise, it should print an error message. Some examples of well-structured strings are:

```
Juventus Milan 4-2
Juventus Roma 2-0
```

8 Tuples

Exercise 1.
Given a tuple `t`:

```
t=('a','b','c','d','e')
```

Decide which of the following stamements are valid tuple operations.

- `t[0]='a'`
- `print t[0]`
- `print t[1:4]`
- `t='a'+t[:]`
- `t=('a',)+t[:]`

Exercise 2.
Write a program which plays the Lottery game between n players as follows: The human player chooses six different numbers in the range 1-100. In the same way, the other $n-1$ players (governed by the program) choose six different numbers in the range 1-100. The program selects 6 different integers at random in the range 1-100. At the end of the game, the program shows the score of each player. Players getting the highest score win the game. The number of players is get in input at the beginning of the game.

Exercise 3.
Write a program which plays the game of "guess the number" as follows: The program chooses the number to be guessed by selecting an integer at random in the range 1 to 1000. The player then types a first guess. The program responds with one of the following answers:
(i) `You guessed the number!`
(ii) `Too low. Try again!`
(iii) `Too high. Try again!`

If the guess of the player is incorrect, the program should loop until the player finally gets the right number. The program should keep telling the player `Too low` or `Too high` to help the player on the correct answer.

Exercise 4.
Write a program which plays a simplified version of the "game of life". The space of the game of Life is a matrix of 40x40 square cells, each of which can be in one of two possible states: live or dead. Each cell interacts with its 8 neighbours, which are the cells that are directly horizontally, vertically or diagonal adjacent. At each step in time, the following transitions occur:

- Any alive cell with fewer than 2 alive neighbours dies, as a consequence of underpopulation;
- Any alive cell with more than 3 alive neighbours dies, as a consequence of overpopulation;
- Any alive cell with 2 or 3 alive neighbours lives, unchanged, to the next generation;
- Any dead tile with exaclty 3 alive neighbours will be populated with a living cell.

At the beginning of the game, a certain number of cells (say K) it is spread out randomly in the matrix. At each step, the transitions (1-4) are applied to all the cells of the system. Then, the system waits to receive an input from user. The rules continues to be applied repeatedly till the user types the character `'Q'`.

9 Dictionaries

Exercise 1.
What is the output of the following program? Use state diagrams if useful.

```
str='hello'
dict={'h':1,'e':2,'l':3}
val=0
for c in str:
    val=val+dict.get(c,-1)
print val
```

Exercise 2.
In the following exercises, we will develop a simple program for the management of orders in a book store. In the management system, each book is characterized by:

- name: the book title

- quantity: the number of stored copies for that book.

Write a function `insertBook()` which:

(i) checks if the book was already present in the book store;
(ii) if it was not, it inserts the book into the system (setting its quantity equal to 1);
(iii) if it was, it increases by one the number of copies.

Use dictionaries to solve the exercise.

Exercise 3.
Write a function `sellBook(titleBook)` which:

(i) checks if the book with title `titleBook` is present in the book store;
(ii) if it is, it decreases by one the number of copies.
(iii) if the number of copies is equal to 0, it removes the book from the system.

Exercise 4.
Write a function `showStore` which prints out the list of books contained in the system. Generate an histogram like this:

```
Ask to the Dust  **** 4
The lord of the ring ** 2
Heart of darkness ********** 10
```

Exercise 5.
What is the output of the following program? Use state diagrams if useful.

```
def fun(d,x):
    if d.has_key(x):
        d[x]=3
dic={'a':1,'b':2,'c':4}
dic2=dic.copy()
dic3=dic
del dic['a']
fun(dic2,'a')
fun(dic3,'a')
dic2['b']=dic2['a']+dic3.get('a',-2)
print dic2['b']
```

Exercise 6.
Write a function `numberOfDays(monthName)` which returns the number of days in a given month `monthName`. For example, `numberOfDays("June")` should produce in output: `30`, while `numberOfDays("NotExistingMonth")` should produce in output: `-1`. Use dictionaries to solve the exercise (assume there are always 28 days in February).

Exercise 7.
By using the function `numberOfDays(monthName)` defined above, write a function `calendar(monthName,firstDay)` which takes as arguments the name of a month, and the name of the first day of the month (for example: `calendar("June","Sunday")`, and returns in output the calendar of the month. The calendar is a dictionary whose entry is given by:

```
(#Month, #Number of the day),
                    #Day of the week
```

For example, `calendar("June","Sunday")` will return:

```
{('June',1): 'Sunday',...,(June,30):
                              'Sunday'}
```

Exercise 8.
By using the function `calendar(monthName,firstDay)` defined above, write a function `calendar2(firstDay)` which takes the name of the first day of the year as argument, and returns in output the calendar of that year.
To solve the exercise, you can combine two dictionaries by using the `update()` of the primary dictionary. For example:

```
>>> d = {'oranges': 3}
>>> ud = {'pears': 4}
>>> d.update(ud)
>>> d
{'pears': 4, 'oranges': 3}
```

10 Files and Exceptions

Exercise 1.
Write a program which keeps reading strings from keyboard till the user types the string "Exit". All the characters read from input should be written in a text file `input.txt`.

Exercise 2.
Write a program which reads the content of `input.txt`, adds a line-counter at the beginning of each line, and writes each line on a new text file `input2.txt`. File `input.txt` should look like this:

```
1: Text1 2: Text2 ...
```

Exercise 3.
By considering the example of the book-store management application (Section 9, Exercises: 2-4) write a function `writeBackup()` which writes the content of the dictionary `store` on a data file `store.dat`.
In the same way, write a function `readBackup()` which reads the data file `store.dat`, loads the content in the dictionary `store` and returns it. Use `pickle` module to preserve data structures.

Exercise 4.
Write a program which reads a text file containing a series of float value, like this:

```
12.04 23.456 15.678 ...
```

, computes the average value and appends it to the end of the file. The program should raise an exception `EmptyFile` if the text file is empty.

Exercise 5.
Write a program which reads from text file `gps.dat` the GPS positions of a set of cars and pedestrians. Each line of the file is structured in this way:

```
OBJECT_ID VALUE_X VALUE_Y
```

where:

- `OBJECT_ID` can be equal to `CAR` or `PEDESTRIAN`;
- `VALUE_X` is a natural number in range 0..30;
- `VALUE_Y` is a natural number in range 0..30

An example of text file is:

```
CAR 12 4 PEDESTRIAN 23 12 CAR 10 10
```

The program should raise an exception `WrongFormat` if the file is not formatted in the right way. Then, the program should display the position of each object on a map 30x30. The character "*" is used to represent the position of a PEDESTRIAN on the map. The character "+" is used to represent the position of a CAR on the map.
Hints: Use `split` method to split a line into tokens. Use dictionaries to keep track of the object positions on the map.

11 Classes and Objects

Exercise 1.
In the following exercises, we will develop a simple program for the simulation of cars' mobility in urban environment. The entities simulated by our program are cars and intersections, governed by traffic lights. Let's assume each traffic light is modelled by a class `TLight` with the following attributes:

- `Position`: the x-coordinate (assume the space is 1-D).
- `State`: red or green

Write a function `newTLight(x,state)` which instantiates and returns a `TLight` object, initialized with the values of `x` and `state`.

Exercise 2.
Write a function `switchState(tl)` which takes a `TLight` object as an argument, and switch its state from "red" to "green" or viceversa.

Exercise 3.
Let's assume each car is modelled by a class `Car` with the following attributes:

- `ID`: the identifier of the car.
- `Position`: the x-coordinate of the car (assume the space is 1-D).
- `Speed`: the current speed of the car.
- `Acceleration`: the current acceleration of the car.

Write a function `safeDistance(tl,car)` which takes a `TLight` object and a `Car` object as arguments and returns `True` if the actual distance between the car and the traffic light is greater than a threshold (assume equal to 20m), `False` otherwise.

Exercise 4.
By using the function `safeDistance(tl,car)` defined above, write a function `move(tl,car,tstep)` which takes a `TLight` object and a `Car` object as arguments and updates the car's position after a `tstep` time interval. Let's assume a car is moving with actual speed v and accelaration a, from position x_0. Then, the new position of that car after a

tstep time interval is given by equations:

$$x(t_{STEP}) = x_0 + v \cdot t_{STEP} + \frac{1}{2} \cdot a \cdot t_{STEP}^2 \quad (4)$$

$$v(t_{STEP}) = a \cdot t_{STEP} + v \quad (5)$$

The values of a and v are defined under these constraints:

- If the distance between the car and the traffic light is lower than a certain threshold, and the traffic light is on "red" state, then the car starts reducing its speed with constant accelation $a = -1m/s$
- Otherwise, if the car's speed is greater than the threshold `maxSPEED` (assume 20m/s) , then the car starts reducing its speed with constant accelation $a = -0.5m/s$
- Otherwise, if the car's speed is lower than the threshold `minSPEED` (assume 10m/s) , then the car starts increasing its speed with constant accelation $a = 0.5m/s$
- Otherwise, if the car's speed is in range [`minSPEED`, `maxSPEED`], then the car keeps moving with a constant speed, i.e. $a = 0m/s$

Exercise 5.
Write a function `createCar(car,newID)` which takes a `Car` object (`car`) as an argument and returns a new `Car` object with identifier `newID` and with the same position, speed and acceleration of the argument `car`. Use `copy` method to duplicate objects.

Exercise 6.
What is the output of the following program? Use state diagrams if useful.

```
import copy class Obj:
    pass
def fun(p):
    p2=copy.copy(p)
    if (p2.b==7):
        p2.b=10
    else:
        p2.a=4
    return p2

p=Obj()
p.a=5
p.b=7
p2=p
p2.b=5
p3=fun(p)
print p3.a + p3.b
```

12 Classes and Functions

Exercise 1.
Define a class `Point`, which represents the coordinates of a point in a 2-D space. Write a function `createPoint(x,y)` which takes as arguments two real numbers and returns a `Point` object.

Exercise 2.
Define a class `Line`, which represents the equation of a line, expressed in the slope-intercept form:

$$y = m \cdot x + n \quad (6)$$

Write a function `passThrough(p,l)` which takes as arguments a `Point` object `p` and a `Line` object `l` and returns `True` if `l` passes through `p`, `False` otherwise. Is `passThrough(p,l)` a modifier function?

Exercise 3.
Write a function `lineThrough(p1,p2)` which takes as arguments two `Points` objects (e.g. `p1` and `p2`) and returns a `Line` object which represents the equation of the line passing through `p1` and `p2`.
Hint: The slope m of the line passing through points (x_1, y_1) and (x_2, y_2) is equal to:

$$m = \frac{y_2 - y_1}{x_2 - x_1} \quad (7)$$

while the intercept n is given by equation:

$$n = \frac{x_2 \cdot y_1 - x_1 \cdot y_2}{x_2 - x_1} \quad (8)$$

Exercise 4.
Define a class `Fraction` for performing arithmetic operations with fractions. The attributes of `Fraction` are:

- the numerator: (an integer)
- the denominator: (an integer)

Write a function `reduce(f)` which takes as argument a `Fraction` object `f` and returns a fraction (i.e. a new `Fraction` object) which is the reduction to lowest terms of `f`. For example, the reduced form of the following fractions:

$$\frac{1}{2} \quad \frac{10}{5} \quad \frac{5}{25} \quad \frac{12}{22}$$

are, respectively:

$$\frac{1}{2} \quad 2 \quad \frac{1}{5} \quad \frac{6}{11}$$

Is `reduce(f)` a pure function or a modifier?

Exercise 5.
Rewrite the function `reduce(f)`, so that it reduces the fraction taken as argument and it stores the result in `f`.
Is `reduce(f)` a pure function or a modifier?

13 Classes and Methods

Exercise 1.
Rewrite the class `Fraction`, by including the function `reduce` (Exercise 12.5) as a method of the class. Call `reduce` to reduce the fraction: $\frac{34}{136}$.

Exercise 2.
Rewrite the class `Line`, by including the function `lineThrough` as an initialization method of the class. Write a new method `printEquation` which prints the equation of the line in the slope-intercept form: (i.e. $y = m \cdot x + n$)
Use `lineThrough` and `printEquation` methods to print the equation of the line passing through points $(23, 35)$ and $(-23, -57)$.

Exercise 3.
Write a method `intersection(self,l)` of the class `Line`, which returns the intersection point (i.e. an object `Point`) between the current line and another line `l`. The method should raise an exception `Parallel` if the two lines are parallel (i.e. they have no intersection point).
Hint: The intersection point between lines $a : y = m_0 \cdot x + n_0$ and $b : y = m_1 \cdot x + n_1$ is given by equations:

$$x = \frac{n_1 - n_0}{m_0 - m_1} \tag{9}$$

$$y = \frac{m_0 \cdot n_1 - m_1 \cdot n_0}{m_0 - m_1} \tag{10}$$

Exercise 4.
Write an initialization method for the class `Fraction`. Override the addition operation `+` to compute the addition of two `Fraction` objects.
Apply the `+` operator to compute: $\frac{3}{4} + \frac{1}{2}$. The result should be reduced to lowest terms.

Exercise 5.
Override the multiplication operation `*` to compute the multiplication of two `Fraction` objects.
Apply the `+` operator to compute: $\frac{5}{4} \cdot \frac{3}{5}$. The result should be reduced to lowest terms.

Exercise 6.
A cluster is defined as a generic set of elements. One of the element of the cluster is called *clusterhead* or *representative* of the cluster. Define a class `Cluster` with the following methods:

- an initialization method, which takes as argument an element x and adds it to the cluster. x becomes also the *clusterhead*.
- a method `setclusterhead(self,x)` which sets x as the clusterhead (if and only if x already belong to the cluster)
- a method `cardinality(self)` which returns the number of elements in the cluster
- a method `insert(self,x)` which adds element `x` to the cluster (if and only if x does not already belong to the cluster)

Exercise 7.
Override the addition operation `+` to compute the addition of two `Cluster` objects. The addition of two clusters `c1` and `c2` produces a third clusters `c3 = c1 + c2`, where:

- the clusterhead of `c3` is the clusterhead of `c1`
- the elements of `c3` are the union of elements of `c1` and `c2` (without duplicates).

Exercise 8.
Override the operation `-` to compute the difference of two `Cluster` objects. The difference of two clusters `c1` and `c2` produces a third clusters `c3 = c1 - c2`, where:

- the clusterhead of `c3` is the clusterhead of `c1`
- the elements of `c3` are the elements of `c1` which are not in `c2` (except for the clusterhead).

14 Set of objects

In section 13, we introduced the concept of "cluster" as a set of elements. More generally,

a cluster is a collection of objects which are similar between them and are dissimilar to the objects belonging to other clusters. A loose definition of clustering could be the process of organizing objects into groups whose members are similar in some way. Clustering algorithms are applied in many fields, including for instance marketing (for finding groups of customers with similar behavior), biology (for the classification of plants and animals given their features), computer science (for the classification of web-site on WWW), and so on. In this section, we will consider a simple application of clustering to partition a set of 2D points. Let's suppose our collection of objects is a set U (Universe), composed by N points, where each point p has real coordinates (x_j, y_j) and $0 \leq x_j \leq 1000$, $0 \leq y_j \leq 1000$, $\forall j \in [1..N]$.

Each cluster C_i is a collection of n points $\{p_i, i \in 1..n\}$. The *centroid* of a cluster is defined as a point $D(x, y)$ whose coordinates are given by equations:

$$x = \frac{x_1 + x_2 + ... + x_n}{n} \quad (11)$$

$$y = \frac{y_1 + y_2 + ... + y_n}{n} \quad (12)$$

Many clustering algorithms have been proposed in the literature.
K-Means is one of the simplest clustering algorithm. It allows to classify a set of data into K clusters (K is known at priori). The algorithm of *K-Means* is composed of the following steps:

1. Create K cluster with no elements. Generate randomly the centroid of each cluster into the 2-D space.

2. Assign each point of U to the cluster that has the closest centroid, according to the Euclidean distance.

3. When all the points have been assigned, recalculate the positions of the centroids for all the K cluster.

4. Repeat Steps 2 and 3 until the centroids no longer move.

At the end of the algorithm, all the points of U are classified into K clusters. It is possible to estimate the "goodness" of the clustering produced by $K - Means$, according to a metric which estimates the similarity of objects contained in each cluster. Such a metric is called "*distortion*" and is defined in this way:

$$D = \sum_{j=1}^{N} \sum_{i=1}^{K} (||c_i, p_i^{(j)}||)^2 \quad (13)$$

where $(||c_i, x_i^{(j)}||)$ is the Euclidean distance between point $p_i^{(j)}$ in cluster i and its centroid c_i. The lower is the value of distortion, the higher is the similarity of objects contained in each cluster.

Exercise 1.
Re-define the class `Point`, so that each `Point` object contains the following attributes:

- a couple of real coordinates;
- a reference to the `Cluster` object containing that point.

Write a method `setCluster(self,c)` which sets the reference to a `Cluster` object `c`.

Exercise 2.
Re-define the class `Cluster`, so that each `Cluster` object contains the following attributes:

- a list of `Point` objects which represents the set of elements of that cluster;
- a `Point` object which represents the centroid of that cluster.

Write the following methods for the class `Cluster`:

- The initialization method, which takes as argument a `Point` object, which is the centroid of the cluster.
- A method `insert(self,p` which takes as argument a `Point` object and adds it to the set of elements of the cluster.
- A method `computeCentroid(self)` which re-computes the coordinates of the centroid of the cluster. It returns `True` if the position of the centroid has moved, `False` otherwise.
- A method `computeDistance(self,p)` which takes as argument a `Point` object `p` and returns the Euclidean distance between the centroid of the cluster and `p`.

- A method `remove(self,p)` which takes as argument a `Point` object and removes it from the set of elements of the cluster.

Exercise 3.
Define a class `Universe`, which contains the initial set of points to be grouped into clusters. Write the initialization method, which takes as argument the number of points N and creates N `Point` objects. The coordinates of each point are randomly generated in the space $[0...1000] \cdot [0...1000]$.

Exercise 4.
Define a class `KMeans` which implements the *K-Means* clustering algorithm. Write the following methods:

- The initialization method, which takes as argument an `Universe` object (i.e. the set of points to be clustered) and a number K (i.e. the number of cluster to create).
- A method `run(self)` which implements the algorithm described above, and returns a list of `Cluster` objects. Moreover, the method should count the number of iterations performed by the algorithm.
- A method `giveIterations(self)` which returns the number of iterations performed by the algorithm.

Exercise 5.
Define a class `Statistics` which prints out some statistics about the clustering process. Write the following methods:

- The initialization method, which takes as argument the list of `Cluster` objects created by the *K-Means* algorithm and the number of iterations performed by the algorithm.
- A method `printCluster(self)` which prints out the content of each `Cluster` objects.
- A method `printStat(self)` which computes the distortion metric, and prints it together with the the number of iterations.

Exercise 6.
Modify the program so that it is possible to perform 100 runs of the *K-Means* algorithm with N=100 (the initial set U is different for each run), while K varies between 5 and 15. Modify the class `Statistics`, so that it keeps global statistics about each run. At the end of runs, the program should produce a table like this:

```
K_VAL Avg_I(k) Avg_D(k)
```

where `Avg_I(k)` is the average number of iterations required by *K-Means* algorithm with K=`K_VAL` and `Avg_D(k)` is the average distortion value.

15 Inheritance

Exercise 1.
Define a class `Employee` with the following attributes:

- `Name`: the Employee's name
- `Age`: the Employee's age (an integer value)
- `Level`: the Employee's level (an integer value). By default, the Employee's level is equal to 1.
- `Working Hours`: the number of working hours in the current month
- `Project`: the name of the project the Employee is working on

Write the following methods for the class `Employee`:

- An initialization method, which takes as arguments the name, age, project and working hours of the Employee.
- A method `printEmployee(self)` which prints all the information about the Employee, e.g.:

```
Name: Rossi
Age: 28
Level: 1
Working Hours: 10
Project: Pegasus
```

- A method `getSalary(self)` which returns the salary of the current month, computed in this way:

```
salary = Level * Working Hours * 20
```

- A method `setProject(self,projectName)` which sets the Employ's project to `projectName`

Exercise 2.
Define a class `Manager` which is a subclass of `Employee`. The `Manager` class extends `Employee` with the following attributes and methods:

- A list of `Employee` supervised by that `Manager`.
- A method `addEmployee(self,employee)` which takes as argument an `Employee` object and adds it to the list of supervised for that `Manager`, if and only if the current level of `employee` is equal to 1. Moreover, it sets the current project of `employee` to the project of the `Manager` object.
- A method `printSupervised(self)` which prints out the information of each `Employee` supervised by that `Manager`.

The level of each `Manager` is equal to 2.

Exercise 3.
Define a class `Director` which is a subclass of `Manager`. Each `Director` supervises a list of `Manager`. The `Director` class extends `Manager` with the following attributes and methods:

- `Budget`: the current budget of the company.
- A method `areSalariesOK(self)` which returns `True` if the total cost for salaries of people working in the company (including simple employees, managers and the director himself) is lower then the current budget, `False` otherwise.
- A method `printProjects(self)` which prints the list of all the projects addressed by the company, and, for each of them, the total number of people working on it.
- A method `printCompany(self)` which prints out the information of all the people working in the company (including simple employees, managers and the director himself).

Moreover, redefine the method `addEmployee(self,employee)` so that it adds an `employee` to the list of supervised for that `Director`, if and only if the current level of `employee` is equal to 2. The current project of `employee` should not be modified. The level of each `Director` is equal to 3.

A sensor is a small device equipped with sensing capabilities and wireless technology. Sensor are used in everyday objects to measure physical quantity such as temperature, humidity, motion, ... There are innumerable applications for sensor of which most people are never aware. In our case, we will consider a simple application scenario, i.e. a fire alert system in an hotel floor. Let's assume our floor is composed by 9 rooms, as shown in the map:

```
*****************
* 0  *  1  *  2 *
*               *
** ***** **** ***
* 3  *  4  *  5 *
*               *
** ***** **** ***
* 6  *  7  *  8 *
*               *
*****************
```

Each room is provided with a sensor which detects the presence of a risk (such as fire eruption). A robot continuously moves from one room to another, and collects data coming from the sensor of the room. Our simulation program works in round. At each round:

- An alert message is produced by one sensor in a room of the floor.
- If the robot is in the same room of the sensor generating the message, the alert message is received, and an emergency procedure is started.
- The position of the robot is updated: according to the movement policy, the robot can decide to stay in a room or to move to an adjacent room.
- Go back to Step 1.

Let's assume at each round one alert message is generated by one of the sensors of the floor. The sensor which generates the message is chosen according to the following distribution:

```
p(0)=p(1)=p(2)=5%
p(3)=p(5)=15%
p(4)=40%
p(6)=p(7)=p(8)=5%
```

Exercise 4.
Define a class `Sensor` with the following methods and attributes:

- An attribute `room` which gives the number of the room (0..8) where the sensor is placed.
- An attribute `message` which counts the number of alert messages generated by that sensor.
- A method `addMessage(self)` which increments by 1 the value of `message`.
- A method `getMessage(self)` which returns the value of `message`.

Exercise 5.
Define a class `MovingSensor` which is a subclass of `Sensor`. The `MovingSensor` class extends `Sensor` with the following attributes and methods:

- A method `getAdjacentRoom(self,roomNo)` which returns the list of rooms adjacent to `roomNo`, including the room itself.
- A method `move(self)` which updates the current position of the robot. The method should be empty in this phase (i.e. the robot does not move)
- A method `misdetect(self)` which is called each time the robot is not in the room where an alert message was generated. The method should be empty in this phase.

In this case, the attribute `message` counts the number of alert messages received by that sensor.

Exercise 6.
Define a class `MovingSensor1` which is a subclass of `MovingSensor`. Redefine the method `move(self)` which updates the current position of the Robot. In this second version, the robot moves randomly, i.e. at each step it decides randomly whether to stay in the current room or to move to an adjacent room.

Exercise 7.
Define a class `MovingSensor2` which is a subclass of `MovingSensor`. The `MovingSensor2` class extends `MovingSensor` with the following attributes and methods:

- An attribute counter `messages)`, which is the total number of alert messages received. In this case, the robot visit each room following a pre-defined sequence:

  `0-1-2-5-4-3-6-7-8-7-6-...`

Exercise 8.
Define a class `MovingSensor3` which is a subclass of `MovingSensor`. Redefine the method `move(self)` which updates the current position of the Robot. Define a "smart" movement policy in order to maximize the number of alert messages detected by the robot.

Exercise 9.
Define a class `AlertSystem` which models the fire alert system in an hotel floor. Write the following methods:

- A method `produceAlert(self)` which returns the number of the room where the alert message is generated. Room is chosen according to the probabilistic distribution stated above.
- A method `run(self,rounds)` which repeats steps 1-4 for a number of interations given by `rounds`.
- A method `statistic(self)` which evaluates the effectiveness of the movement policy, based on this metric:

$$e = \frac{\#Alert_Message_Received}{\#Rounds} \tag{14}$$

  and prints out the result.

Create instances of `MovingSensor1`, `MovingSensor2` and `MovingSensor3`. Run the simulation with `rounds=100`. Which movement policy provides the highest efficiency?