Fundamentals of Programming – Partial exam (example)

Important: Provide the best possible solutions in each exercise. In addition to working properly, procedures and functions must be well programmed (using the most appropriate instructions, without unnecessary operations or variables, etc.).

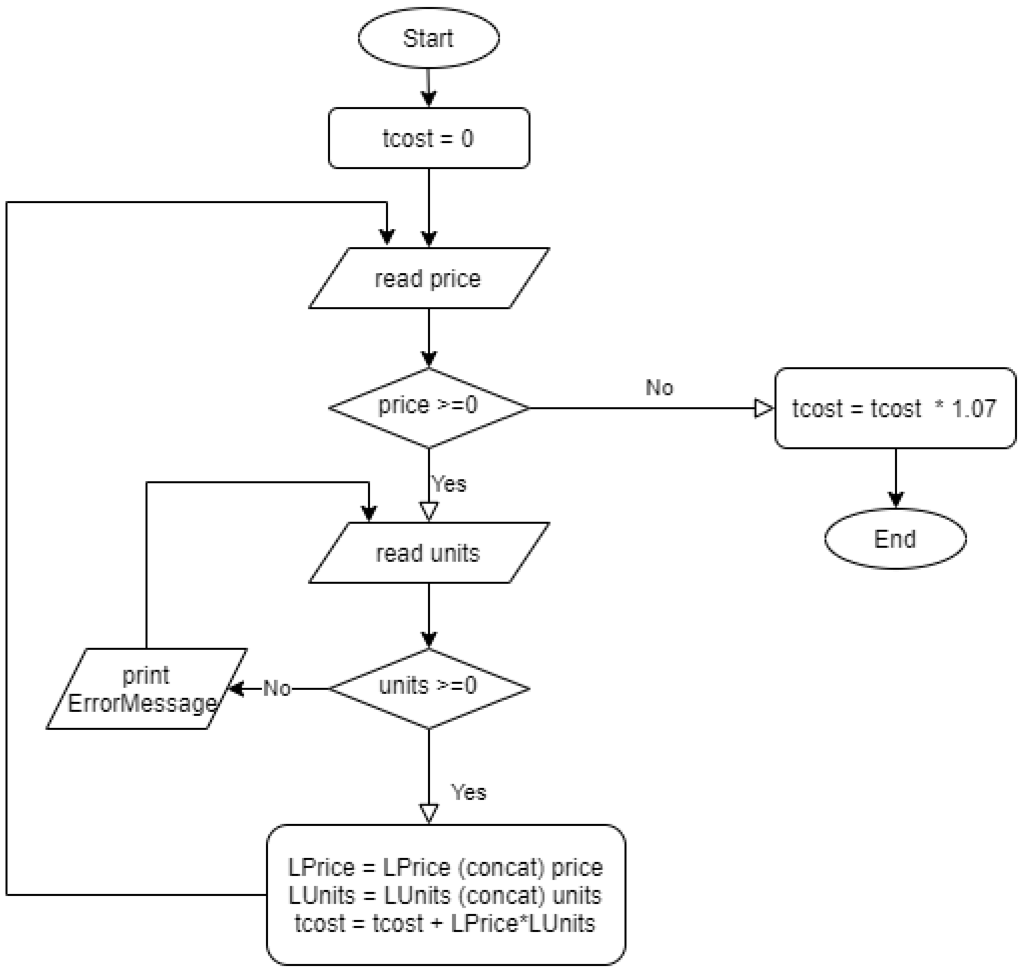
Exercises 1 to 6 refer to the same context, introduced below:

We want to make a program to make the sales invoicing of a company. The program must ask for the data of a customer and the data of the products that have been sold to calculate the total amount of the sale. It must also apply discounts based on the amount of the sale and make histograms.

Exercise 1 (1 point)
Make the flowchart of a function that calculates the total amount of a customer sale. It will ask the user to introduce the price per unit, and then, the number of units sold of the product. The price per unit will be stored in the list *LPrice*, whereas the units will be stored in the list *LUnits*. This process will be repeated for each product on the sale. When a negative price is entered, the sale ends.

If the user enters a negative number of units sold, an error message is displayed ("Error: units cannot be negative") and it should ask again the amount of units until a positive integer number is introduced.

Finally, a 7% VAT is applied to the total amount of the sale.



Exercise 2 (1 point)
Write the python code of the previous exercise, and name the function as `sale()`. The function returns the amount of the sale.

```
def sale():
    tcost=0
    LPrice = []
    LUnits = []
    price=float(input("Price per unit: "))
    while price>=0:
        units=int(input("Units: "))
        while units < 0:
            print("Error: num of units cannot be negative")
            units=int(input("Units: "))
        tcost+=price*units
        LPrice.append(price)
        LUnits.append(units)
        print(tcost)
        price=float(input("Price per unit: "))

    #print(LPrice)
    #print(LUnits)
    #print(tcost)
    tcost+=tcost*0.07
    return tcost
```

Exercise 3 (1 point)
Write the function `discount()` to calculate the discount to be applied to customers depending on the amount of their sale. This function will receive a real value corresponding to the amount of the sale and it will return the amount to be paid by the customer. The discount that will be applied is:

- 0% if the amount of the sale is less than 500€.
- 1% if the amount of the sale is in the range [500€, 1000€).
- 2% if the amount of the sale is in the range [1000€, 2000€).
- 3% if the amount of the sale is equal to or greater than 2000€.

Note: Consider that the amount will never be negative (no need to check that).

```
def discount(cost):

    if cost < 500:

        return cost

    elif cost < 1000:

        return cost-cost*0.01

    elif cost < 2000:

        return cost-cost*0.02

    else: # 2000 or more

        return cost-cost*0.03
```

Exercise 4 (1 point)
Make a function called `file2list()` that takes as a parameter the name of a file (string) that contains (numeric) barcodes of products, and returns a list with the elements that are read. Each line in the file contains one barcode and ends using the line break (‘\n’).

Important: Use exceptions to control that the file can be opened. If the file is not found, make a `raise` with the `FileNotFoundError`.

```
def file2list(file_name):
    list_barcodes=[]
    try:
        f=open(file_name,"r")
    except:
        raise FileNotFoundError
    else:
        for line in f:
            list_barcodes.append(int(line[:-1]))
        f.close()
        return(list_barcodes)
```

Exercise 5 (1 point)
Make a function named `makeHistogram()` that receives a list with the amount of the invoices of the company during the year. It will count how many invoices are below 1000€, how many between [1000€, 2000€) and how many above 2000€.

At the beginning, create the list *HistogramSales* and initialize it with zeros. The function will return the histogram.

```
def makeHistogram(list_invoices):
    hist = [0,0,0]
    for i in list_invoices:
        if i<1000:
            hist[0]+=1
        elif 1000<i<2000:
            hist[1]+=1
        else:
            hist[2]+=1
    return hist
```

Exercise 6 (2 points)
Make a main program that follows these steps:

1. Ask the user the name of the file with the codes of products, and call the function *file2list()* to read the file to obtain the list of products.
2. Print the menu, using the procedure `print_menu()`, which we will assume that it is already implemented:
    a. Make an invoice
    b. Apply discount
    c. Compute the histogram of sales
    d. Exit
3. If the user enters option **a)**
    3.1 Call function `sale()` to introduce the data of the sale and compute the cost.
    3.2. Print the amount of the sale
4. If the user enters option **b)**
    4.1. Ask the user the amount of the sale and call the function `discount()`.
    4.2. Print the final amount, after applying the discount.
    4.3. Save the final amount into a list called `ListInvoices`.
5. If the user enters option **c)**, call the function `makeHistogram()` and print the histogram.
6. If the user enters option **d)**, the program ends.
7. If the user enters any other option, it will show the message “Error, option does not exist” and will go back to option 2.

```
def print_menu():
   print(" MENU \n a)Make invoice \n b)Apply discount \n c)Compute histogram \n d)Exit")

###############################     MAIN   #####################
filename = input("Write name file:")
try:
   lbarcod = file2list(filename)
except FileNotFoundError:
   print("Error: File not found!")
else:
   print(lbarcod)

ListInvoices = []
print_menu()
option = input("Option:")
while option != "d":
   if(option=="a"):
        cost=sale()
        print("Cost of sale:",cost)
   elif(option=="b"):
        thecost = int(input("Cost of sale:"))
        fcost = discount(thecost)
        print("Final cost (after discount):",fcost)
        ListInvoices.append(fcost)
   elif(option=="c"):
        h = makeHistogram(ListInvoices)
        print(h)
   else:
        print("Error, option does not exist")
   print_menu()
   option = input("Option:")
```

Exercise 7 (1’5 points)

Make a program that asks for a string and prints each word in a different line, but in uppercase. Consider that all words in the string are separated by the blank space. Punctuation marks are not required.

For example, if the string you enter is "Hello, how are you?", it should print:

```
HELLO,
HOW
ARE
YOU?
```

```
s=input("Write a string: ")
i=0
while i<len(s):
   word=""
   while i<len(s) and (s[i]!=" "):
      word+=s[i]
      i+=1
   i+=1
   if (word !=""):
      print("WORD:", word.upper())
```

Exercise 8 (1’5 points)

What will be the values of x, y and z at the end of the execution of the program?

```
def process(x,w,lista):
    for i in range(w):
        lista.append(x)
        x+=w
    return x


def f(x,y,z):
    lista=[]
    z=process(x,y,lista)
    x=len(lista)
    y+=1
    return(z,x,y)


x,y,z = f(2,3,2)
```

Value x	Value y	Value z
11	3	4