BIRKBECK

(University of London)

MSc EXAMINATION FOR INTERNAL STUDENTS

MSc Computer Science

MSc Data Science

Department of Computer Science and Information Systems

Principles of Programming I

BUCI033S7

DATE OF EXAMINATION: Monday, 30th April 2018

DURATION OF PAPER: One Hour

Practical — Mock Paper — Evening

WITH OUTLINE SOLUTIONS

RUBRIC:

1. Candidates should attempt ALL 4 questions on this paper.
2. You are advised to look through the entire examination paper before getting started, in order to plan your strategy.
3. Simplicity and clarity of expression in your answers is important.
4. All programming questions should be answered using the Python programming language.
5. Electronic calculators are **NOT** allowed.
6. Start each question on a new page.

Question:	1	2	3	4	Total
Marks:	7	10	3	30	50

Question 1 ..... Total: 7 marks

Write a function to search a list of strings for a specified value. The function takes a search string, `str`), a list of strings (`lst`), and the size of the list (`size`), and returns the number of the slot that contains the search string, or returns -1 if the search string does not appear.

```
search (str, lst, size)
```

Solution:

```
def search (str, lst, size):
    position = -1
    for x in range(0,size):
        if str == lst[x]:
            position = x
            break
    return position
```

Question 2 ..... Total: 10 marks

Write a program for a number guessing game. The program generates a random number between 0 and 99, and then asks the user to guess that number. For each guess the program replies `Correct`, `Too low`, or `Too high`. If the number is correct, the program prints the number of guesses it took. If not, the program asks the user to guess again. For example:

```
Guess a number between 0 and 99: 50
Too low. Guess again: 75
Too high. Guess again: 60
Too high. Guess again: 54
Correct. It took you 4 guesses.
```

Your program should make use of a boolean *flag*.

Solution: 3 marks for the loop control including the flag.
1 marks for correctly calling the random function.
2 mark for counting. This time they have to start the counter at 1 when the first guess is made, then add one as each guess is read.
The other 4 marks are for requesting a guess, reading the input, converting it, testing it, and taking appropriate action.

```
from random import randrange

number = randrange(0,100)
guess = int(input("Guess (0..99): "))
count = 1
wrong = guess != number
while wrong:
```

```
    if guess < number:
        print("Too low")
    else:
        print("Too high")
    guess = int(input("Guess again: "))
    count = count + 1
    wrong = guess != number

print("Correct. It took you ", count, " guesses.")
```

Question 3 ..... Total: 3 marks

Complete the function definition so it returns the square of the product of the parameters, so `sqrProd(2, 5)` returns `(2*5)*(2*5) = 100`.

```
def sqrProd(x, y):
```

Solution:

```
def sqrProd(x,y):
        sq = x * y
        return sq * sq
```

or similar.

Question 4 ..... Total: 30 marks

A mobile telephone company offers two types of service: *regular* and *premium*. Its rates vary, depending on the type of service. The monthly rates are computed as follows:

Regular service:	£10.00 for the first 50 minutes. Charges for over 50 minutes are 20 pence per minute.
Premium service:	£25.00 monthly charge plus a. for calls made from 6:00am to 6:00pm (daytime), the first 75 minutes are free; charges for over 75 minutes are 10 pence per minute b. for calls made from 6:00pm to 6:00am (off-peak), the first 100 minutes are free; charges for over 100 minutes are 5 pence per minute.

Write a program to calculate and print mobile phone bills. The input is a series of records each giving the account number (a string), the customer surname, the balance on the account from last month, the type of service (`R` or `P`, for `R`egular or `P`remium, as explained below), followed by a single number of call minutes in the case of Regular service, and the number of daytime minutes and the number of off-peak minutes in the case of Premium customers.

The file is terminated by a value of `X0000` in place of an account number. A mini example input file, containing one of each type of record is

```
A9845   Hurley  37.35 R 55
A9846   Hicks   0.00  P 70 139
X0000
```

Your program should output, for each input record the account number, name, the type of service, the charge for the month, and the new balance on the account.

Your program must declare, define and use functions to read each record, to compute regular charges, to compute premium charges, and to print each output record.

Solution:

```
# More than would have been expected as (much of) the code has been
# refactored to make "best use" of functions.

END = "X0000"
REGULAR = "R"
PREMIUM = "P"

def readRec():
    return input()

def computeRegular(mins):
    cost = 10.00
    limit = 50
    charge = 0.2

    rest = compute(mins, limit, charge)

    return cost + rest

def compute(mins, limit, charge):
    cost = 0
    remain = mins - limit
    if remain > 0:
        cost = remain * charge
    return cost

def computePremium(daytime,offpeak):
    cost = 25.00
    charge = 0.1
    peakCharge = 0.05
    daytimeLimit = 75
    offpeakLimit = 100

    # compute daytime
    rest = compute(daytime, daytimeLimit, charge)
```



```
    cost += rest

    # compute off-peak
    rest = compute(offpeak, offpeakLimit, peakCharge)
    cost += rest

    return cost

def formatLine(account, name, kind, bill, balance):
    return "Account: " + account + " Name: " + name + "\t" + \
           kind + "\t" + bill + " " + balance

def printLine(line):
    print(line)

if __name__ == "__main__":
    billings = []
    line = readRec().split()

    while (line[0] != END):
        if(line[3] == REGULAR):
            bill = computeRegular(int(line[4]))
            kind = "regular"
        else:
            bill = computePremium(int(line[4]),int(line[5]))
            kind = "premium"
        newBalance = float(line[2]) - bill
        billingLine = formatLine(line[0], line[1], kind, str(bill),
                                 str(newBalance))
        billings.append(billingLine)
        line = readRec().split()

    for x in billings:
           printLine(x)
```