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

WITH OUTLINE SOLUTIONS

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:	13	7	10	20	50

Question 1 . . . . Total: 13 marks

The Fibonacci series starts with $0, 1, \ldots$; each term is the sum of the two previous terms. For example,

```
0 1 1 2 3 5 8 13 ...
```

(a) Write a Python program to print out the first 20 fibonacci numbers, each number separated by a space. $\boxed{\text{7 marks}}$

Solution: 3 marks for a correct loop (one mark docked if the number of iterations was off by one or two); 3 marks for the body of the loopprinting and shifting terms; one for printing out the first two terms.

```python
prev = 0
curr = 1
print(prev, curr, end=" ")
for x in range(3, 20 + 1):
    next = prev + curr
    print(next, end=" ")
    prev = curr
    curr = next
```

(b) Write a second Python program to print out the fibonacci series, separated by spaces, up to, but not including, the number 2000, and then print out how many terms were printed. $\boxed{\text{6 marks}}$

Solution: The whole point of this question is to use a WHILE loop keep printing fibs while the one you are about to print is less than or equal to 2000.

The main challenge is ensuring that your WHILE loop condition comes at the right time.

3 marks for the counting bit – initialising counter (to 2), incrementing it and printing it.
3 marks for the while loop – using one, testing the number before you print it.
No marks for the basic Fibonacci computation, since that was covered in part (a).

```python
prev = 0
curr = 1
print(prev, curr, end=" ")
# two terms printed so far
counter = 2
next = prev + curr

while next <= 2000:
    print(next, end=" ")
    counter = counter + 1
    # shift the terms
```

Outline Solutions

```
        prev = curr
        curr = next
        next = prev + curr

    print()
```

Question 2 . . . . 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 3 . . . . 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.

Outline Solutions

```
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 4 . . . . Total: 20 marks

Write a program to score True-False tests. Correct answers get 1 point, wrong answers -1 point and no answer (i.e., `x`) gets 0. The program first reads in the thirty correct answers to the test questions (each answer being `T` or `F`). It then reads a series of answer sets – each of these is a student number (an integer) followed by thirty answers, each of which is one of the characters `T`, `F`, or `x`. `x` is used to report that no answer was given. The input is terminated by a record beginning with the student number 999. The program should output each student number followed by their score on the test.

Use functions as appropriate.

Sample input:

```
    T F T F T F T F T F T F T F T F T F T F T F T F T F T F T F
100 T F T T F F T F T F T F T F T T T F F F x F T x T F T F x F
101 T T T T T T T F T F x x x x x x T F T F T F T F T F F F F T
110 T F T F T F T F T F T F T F T F T F T F T F T F T F T F T F
999
```

Sample output:

```
100 19 marks
101 12 marks
110 30 marks
```

Solution:

```
# Not necessarily the best solution but something we would expect
# given the knowledge from the module
```

Outline solutions

```
SENTINEL = 999
TESTSIZE = 30
CORRECT = 1
WRONG = -1
NOTHING = 0
NOANSWER = "x"


def readSolutions(solutions, TESTSIZE):
    # read solutions line with TESTSIZE entries
    str = input()
    solutions = str.split()
    assert len(solutions) == TESTSIZE
    return solutions

def readAnswers(TESTSIZE):
    # read student id + answers --- horrible duplication
    str = input()
    if len(str) != 0:
        answers = str.split()
    studentID = int(answers.pop(0))
    return (studentID, answers)

def score(answers, solutions, TESTSIZE):
    # how many correct/incorrect
    score = 0
    for item in range(TESTSIZE):
        if answers[item] == solutions[item]:
            score += CORRECT
        else:
            if answers[item] != NOANSWER:
                score += WRONG # yes, it is adding a minus number
        # no answer = no mark
    return score


if __name__ == "__main__":
    solutions = []
    grades = {}

    solutions = readSolutions(solutions, TESTSIZE)

    (studentID, answers) = readAnswers(TESTSIZE)
    while studentID != SENTINEL:
        grade = score(answers, solutions, TESTSIZE)
        grades[studentID] = grade
        (studentID, answers) = readAnswers(TESTSIZE)

    for k,v in grades.items():
        print(k, v, "marks")
```

Outline solutions