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: SOMEDAY 2018
TIME OF EXAMINATION: SOMETIME
DURATION OF PAPER: Two Hours

One hour practical (online) + One hour written.

RUBRIC:

1. Candidates should attempt ALL 10 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.

Part A — Practical

Question:	1	2	3	4	Total
Marks:	13	7	10	20	50

Part B — Written

Question:	5	6	7	8	9	10	Total
Marks:	10	5	5	12	6	12	50

 ©Birkbeck 2018

Part A — Practical

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. [7 marks]

(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. [6 marks]

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)
```

Question 3 ..... Total: 10 marks

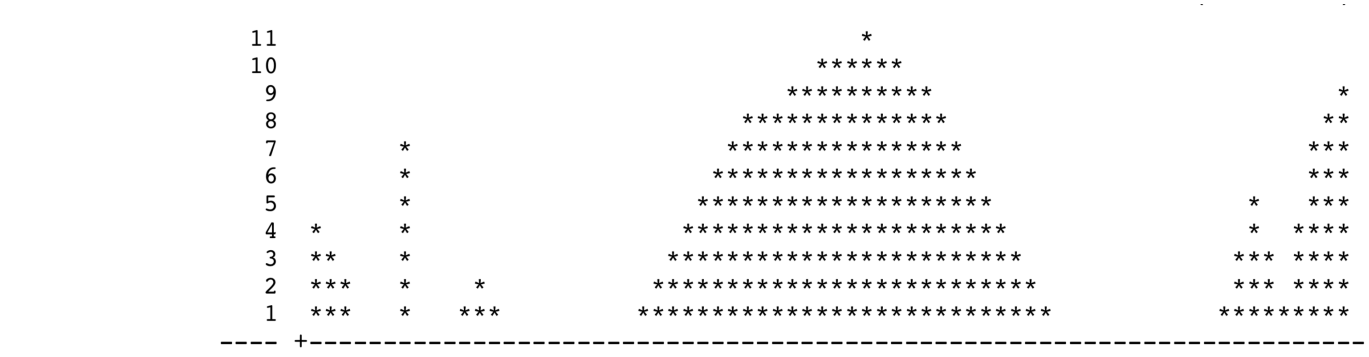
Write a program for a number guessing game. The program generates a random number between 0 and 99, and then asks the user to guess that number. For each guess the program replies `Correct`, `Too low`, or `Too high`. If the number is correct, the program prints the number of guesses it took. If not,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

Question 4 ..... Total: 20 marks

Write a program to read n positive integers and represent them as a bar chart like the one below which plots 4, 3, 2, 0, 0, 0, 7, etc. The y-axis should extend up only as far as necessary.



 ©Birkbeck 2018

Part B — Written

Question 5 ..... Total: 10 marks

(a) In a method inside a class, what important value is held in the variable `self`? Provide an appropriate example to illustrate you answer. [2 marks]

(b) Inside a function `foo` you see the statement `print(bar)`. How do you determine whether `bar` is a *local* or *global* item? Provide a rule, or rules, for determining this. [4 marks]

(c) Consider the following method (defined in a class `WordPlay`): [4 marks]

```
def isVowel(self, ch):
        return ch in "aeiouAEIOU"
```

Write a complete `PyTest` test method for the `isVowel` method, testing positive and negative cases.

Question 6 ..... Total: 5 marks

If the function `f` is defined as:

```
def f(x, y):
    if x == y:
        return 0
    else:
        return f(x - 1, y) + 2 * 2
```

(a) For each of the following what value is returned by the call? [2 marks]

i. `f(2,2)`

ii. `f(7,3)`

(b) What happens if `x` is less than `y` (e.g., `f(1,3)`)? [3 marks]

Question 7 ..... Total: 5 marks

Integer multiplication can be performed using the idea of repeated addition. For a positive integer `n`:

```
m * n = m * (n-1) + m
m * 0 = 0
```

Write a recursive function:

```
multiply(m, n)
```

which multiplies `m` by `n` using repeated addition. Do not use the multiplication operation "`*`" in your answer.

Question 8 ..... Total: 12 marks

Given an integer n, produce a two-dimensional array of size $(n \times n)$ and set each value according to the following rules:

- On the main diagonal set the value to 0.
- On the diagonals adjacent to the main diagonal, set the value to 1.
- On the next adjacent diagonals set the value to 2, and so on.

You should print out the elements of the resulting array with a space between each element and a line break at the end of each row.

For example, for $n = 5$, the resulting array would have the following values:

```
0 1 2 3 4
1 0 1 2 3
2 1 0 1 2
3 2 1 0 1
4 3 2 1 0
```

You should not simply assign to each element in the array but should utilise appropriate iteration constructs.

Question 9 ..... Total: 6 marks

Given a sequence of numbers, determine if the next number has already been encountered. For each number, print the word `YES` (on a separate line) if this number has already been encountered, and print `NO`, if it has not already been encountered. You should employ *sets* in your answer.

For example, if given the following sequence as input:

```
1 2 3 2 3 4
```

the output should be

```
NO
NO
NO
YES
YES
NO
```

Question 10 ..... Total: 12 marks

You are given the following input:

```
3
apple orange banana banana orange
banana grapefruit banana apple orange
orange grapefruit banana apple
```

Where the first line contains the number of lines of input and the successive lines contain a list of words. Print the word in each line of text that occurs most often. If there are several such words, print the one that is less in the alphabetical order.

For the sample input the output would be:

```
banana
banana
apple
```

 ©Birkbeck 2018