Habib University
CS 110 - Computational Thinking I
Fall 2015

Programming Problems 1
Assigned: 21 Sep, Due: 5 Oct, 18h

– This Problem Set has 7 questions for a total of 110 points.
– Submission is open until the indicated date.
– Submit all your files using `svn`.

Problem 1. SVN [30 points]

(a) $\boxed{\text{5 points}}$ Log in to Habib University's SVN server at `http://scm.sse.habib.edu.pk`.

(b) $\boxed{\text{5 points}}$ *Checkout* your *repository* by clicking on it in SCM and following the Checkout instructions. You may be asked for your password.

(c) $\boxed{\text{5 points}}$ There will now be a sub-directory named `<your_user_name>` in your current directory. Change to it and create a new directory in it called `ct1`.

(d) $\boxed{\text{5 points}}$ Mark the `ct1` subdirectory for *addition* to your SCM repository. Files that have been *add*'ed are said to be *tracked*.

(e) $\boxed{\text{5 points}}$ In the `ct1` directory, create a directory called `problems1` and change to it. Save all your programs in this directory. Make sure to *add* every new file and directory. All tracked files and directories will eventually be uploaded to your SCM repository.

(f) $\boxed{\text{5 points}}$ Upload all tracked files and directories to your SCM repository by performing a `commit`. You may be asked for your password. Make sure to `commit` every time you want to save your edits to your SCM repository. You will be asked for a *commit message*. This is a short summary of the changes included in this commit. Make sure to commit your files before you leave the lab today. Otherwise, you will lose your edits.

Problem 2. Hello World [15 points]

(a) $\boxed{\text{5 points}}$ Write a program that prints `Hello world!` to screen. Save your program to a file, `hello.py`.

(b) $\boxed{\text{5 points}}$ Write a program that prints `Hello world!` 50 times to screen. Each `Hello World!` should appear on its own line. Save your program to a file, `hello50.py`.

(c) $\boxed{\text{5 points}}$ Write a program that asks the user how many times to print `Hello world!` and prints that many `Hello World!`s, each on its own line. Save your program to a file, `hello-input.py`.

Problem 3. Academic Performance [20 points]

Grade	Points
A+	4.00
A	4.00
A-	3.67
B+	3.33
B	3.00
B-	2.67
C+	2.33
C	2.00
C-	1.67
F	0.00

Points	Grade
4.00	A+
$(3.67, 4.00)$	A
$(3.33, 3.67]$	A-
$(3.00, 3.33]$	B+
$(2.67, 3.00]$	B
$(2.33, 2.67]$	B-
$(2.00, 2.33]$	C+
$(1.67, 2.00]$	C
$(0.00, 1.67]$	C-
0.00	F

(a) $\boxed{\text{5 points}}$ Write a program that inputs the user's name, e.g. `Habib`, and outputs a greeting, e.g. `Hello, Habib!`, to screen. Save your program to a file, `hello-user.py`.

(b) $\boxed{\text{5 points}}$ Write a program that inputs the number of courses the user is taking this semester. Your program should then take as input the name and credit hours of each of those course and the grade that the user received in each course. Save your program to a file, `courses.py`.

(c) $\boxed{\text{5 points}}$ Write a program that does all of the above and then computes and outputs the user's GPA. The output should be of the following format.

```
Hello, Habib!
You took 6 courses this semester.
Following are the names, credits, and your grade in
    each.
1. CORE101 Rhetoric and Communication - 5 - A-
2. CS110 Computational Thinking I - 4 - A
3. CSD103 Visual Communication - 3 - B+
4. SDP101 Development and Social Change - 3 - A-
5. PUNJ101 Intordotion to Punjabi - 3 - A+
Your gpa is 3.7417.
```

Save your program to a file, `gpa.py`.

(d) $\boxed{\text{5 points}}$ Write a program that does all of the above and outputs an additional line containing the user's letter grade corresponding to the GPA. The letter grade for a gpa is as given in the second table on the left. For the example above, the additional line would be,

```
Your grade is A.
```

Save your program to a file, `grade.py`.

Problem 4. Arithmetic [10 points]

(a) $\boxed{\text{5 points}}$ Write a program that inputs two numbers, x and y, and outputs the product, $x * y$. Your program should compute the product using repeated addition only. Save your program to a file, `product.py`.

(b) $\boxed{\text{5 points}}$ Write a program that inputs two numbers, x and y, and outputs the power, x^y. Your program should compute the power using repeated multiplication only. Save your program to a file, `power.py`.

Problem 5. Box [15 points]

Each program below will input two numbers, `width` and `height` and will output a box. The type of box to be output varies in each part. The examples below assume `width = 4` and `height = 6`.

(a) $\boxed{\text{5 points}}$ Draw a filled box, e.g.

```
****
****
****
****
****
****
```

Save your program to a file, `filled.py`.

(b) $\boxed{\text{5 points}}$ Draw an empty box, e.g.

```
****
*  *
*  *
*  *
*  *
****
```

Save your program to a file, `unfilled.py`.

(c) $\boxed{\text{5 points}}$ Draw an empty square that uses the value of `width` only, e.g.

```
****
** *
* **
****
```

Save your program to a file, `diagonal.py`.

Problem 6. The Grid [10 points]

```
+ - - - - + - - - - +
|         |         |
+ - - - - + - - - - +
|         |         |
+ - - - - + - - - - +
|         |         |
+ - - - - + - - - - +
```

Write a program that takes user input for:
– number of rows,
– number of columns,
– width of a column, and
– height of a row.
and prints a grid accordingly. For each input, allow a default value of 4, i.e. if the user presses Enter without entering a value, 4 should be used as the value for that input. The example on the right shows the grid for
– number of rows = 3,
– number of columns = 2,
– width of a column = 4,
– height of a row = 1.
Save your program to a file, `grid.py`.

Problem 7. Pet the cat [10 points]

Write a menu driven program which helps you take care of your virtual cat. The cat should have a name and age which your program should input. You should be able to do the following actions: 1. Feed it, 2. Pat it, 3. Play with it, and 4. Quit. There should be levels that keep track of how hungry your cat is, how loving it is, and how playful it is. These are set to 50 when the program starts and change dependinig on the user's actions as the program proceeds. When the cat is fed, it becomes less hungry, but it also becomes less playful and less loving. Patting does not affect hunger, increases playfulness and makes the cat more loving. Playing increases hunger and makes the cat more loving but tires the cat so its playfulness decreases. When the user quits, the program should output the current state of the cat. Save your program to a file, `cat.py`.