

ASSIGNMENT I

Assigned: Wednesday, February 3rd, 2016

Due: on or before February 17th, 11:59 pm.

Write a program in each of the following three languages:

(60 points – 20 per language)

- i) Python
- ii) Java
- iii) SML

Consisting of a function that accepts two strings. Each string is the name of a file. The first is the name of an input file and the second is the name of an output file. Name the function 'hw1'. (Note that your program can also make use of other helper functions – just make sure function 'hw1' takes as arguments the input file and output file that are specified in the program)

A pangram is a sentence that contains all the letters of the English alphabet at least once. For example, *the quick brown fox jumps over the lazy dog* is a pangram. The program you are to write must read in an input file(input.txt - a plain text file which contains 5 sentences), line by line and check if the line read is a pangram or not. If the sentence read is a pangram, it writes 'true' to the output file. If it's not, it writes 'false' to the output file.

For example, if input.txt contains:

we promptly judged antique ivory buckles for the next prize.

how quickly daft jumping zebras vex.

pottery is an art.

crazy fredrick bought many very exquisite opal jewels.

mr.dumbledore is a funny name for a dog.

Then your program must output to the output file:

true

true

false

true

false

NOTE : Output is case sensitive – please use all lower case in the output file. **The example provided here is formatted to human readability. Please look at the sample input output files to see the precise formatting.**

You can assume that input.txt contains exactly 5 sentences and all the letters are in lower case. Please use the sample test cases provided to test your code. For the purpose of this assignment you do not need to do any specific error checking on the files. Your program can assume that the files exist (for the input file) or can be created or overwritten (for the output file).

Put your answers for Python, Java and SML in files named, respectively:

hw1.py

hw1.java

hw1.ml

Typo! This should be hw1.sml

PART I – PYTHON AND SML

Additional resources for python:

Skeleton code:

```
def hw1(input, output):
```

```
    #read input file line by line
```

```
    #for every line in input file, check if it contains all alphabets – store result  
    (true/false) in a variable
```

```
    #write result to output file
```

Modification 2/8/16

This last line is the function call to hw1 to test your code. **Don't include this in your submission but use this line to test your code**

```
hw1('input.txt','output.txt')
```

Refer these for additional help:

Section 7.2 – Reading and Writing Files - <https://docs.python.org/2/tutorial/inputoutput.html>

For loops, Lists in Python - <http://learnpythonthehardway.org/book/ex32.html>

Additional resources for Python can be found on Piazza.

Additional resources for SML:

Skeleton code:

```
fun hw1(inFile : string, outFile : string) =
```

(* Open input and output file streams, read first input line from the input file stream *)

(* To read subsequent lines, you can make use of a helper function *) – see resources on SML File Stream

(* Convert the string type of the lines read and alphabet string to list type*)

(* Perform alphabet checks on these lists *)

(*Write the resultant value – true or false to the output file *)

(* Remember to flush the outStream – refer resources on TestIO to see how flushOut is done. *)

(* This last line is the function call to hw1 to test your code. **Don't include this in your submission but use this line to test your code** *)

```
val _ = hw1("input.txt", "output.txt")
```

SML File Stream resources :

Modification 2/8/16

You can use TextIO.openIn and TextIO.openOut for the file stream input and output.

To read each line from a file, you can use the TextIO.inputLine function. This function returns a string option type. (Note that 'string' type and 'string option' type are different). The options it returns are either NONE or SOME(c) depending on content of the line read. If a line is read successfully, it returns SOME(c) and if there is no more data to read, it returns NONE. So to keep reading the file line by line, your code has to keep reading as long as there is SOME(c). To see how it works, refer to the last example in the following link:

<http://www.cs.cornell.edu/courses/cs312/2006fa/recitations/rec09.html> (See the last Example using TextIO)

Resources on Strings and Chars :

<http://sml-family.org/Basis/string.html#SIG:STRING.char:TY>

<http://sml-family.org/Basis/char.html#SIG:CHAR.char:TY:SPEC>

PART II : JAVA

In the folder 'java', create an input.txt file containing any one of the sample input test cases. Your code should be written in hw1.java

Create a 'hw1' class with a static method hw1() that takes in two strings as arguments – first for the name of input file and second for the name of output file. Your function definition goes inside the static method hw1().

You should have another class with a main method to test your code in a separate file. This file shouldn't be submitted – it is for your testing purposes only.

For instance:

Modification 2/8/16

Define hw1.java as :

```
public class hw1{
    public static void hw1(inFile, outFile){
        //function definition here
    }
}
```

Define Tester.java as: (this is a separate file)

```
public class Tester{
    public static void main(String[] args){
        ...
        String input = "input.txt";
        String output = "output.txt";
        /* You can also read input and output as command line arguments – it doesn't
        matter as long as your input file is called input.txt and output file is output.txt */

        hw1.hw1(input,output)
    }
}
```

Modification
2/15/16

If you are using Eclipse or any other IDE's make sure you **do not create any packages** since this will not work when your code is moved to Timberlake (unless you have the appropriate directories created) or on the testing script we use. To test your code on Timberlake, you can use the following commands:

javac Tester.java (This should create a class file in the directory called Tester.class.)

java Tester (without the .class extension)

If executed correctly, your output file should now be populated with the test case results.

Instructions on how to submit the homework:

Create a folder UBITName_HW1 that contains three sub folders Python, SML and Java. The 'Python' folder should contain the following files :

hw1.py

Similarly 'SML' folder contains:

hw1.sml

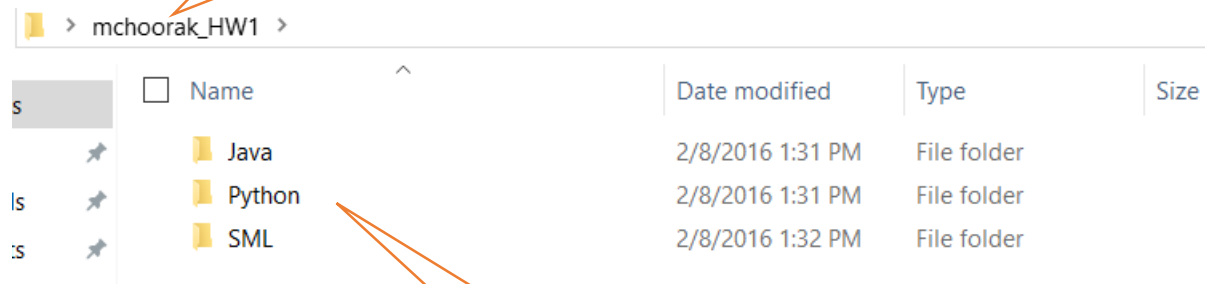
And 'Java' folder contains:

hw1.java

Modification 2/8/16

For instance,

Name of your root folder.
yourUBITname_HW1

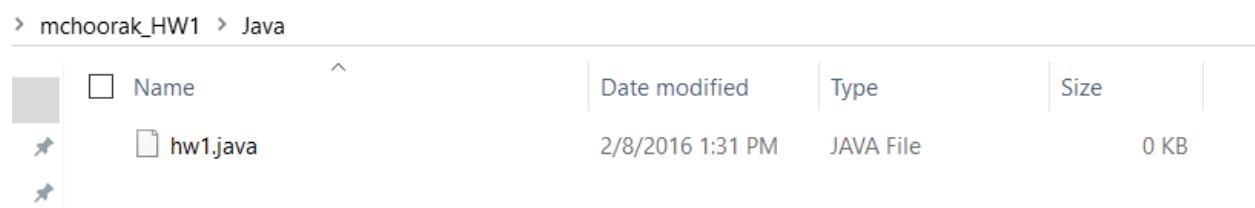


The screenshot shows a file explorer window with the address bar set to 'mchoorak_HW1 >'. Below the address bar is a table listing the contents of the folder. The table has columns for 'Name', 'Date modified', 'Type', and 'Size'. There are three entries, all of which are folders: 'Java', 'Python', and 'SML'. The 'Python' folder is highlighted with an orange oval and a callout pointing to it from the text 'Sub folders inside your folder'.

Name	Date modified	Type	Size
Java	2/8/2016 1:31 PM	File folder	
Python	2/8/2016 1:31 PM	File folder	
SML	2/8/2016 1:32 PM	File folder	

Sub folders inside your
folder


In Java folder,




The screenshot shows a file explorer window with the address bar set to 'mchoorak_HW1 > Java'. Below the address bar is a table listing the contents of the folder. The table has columns for 'Name', 'Date modified', 'Type', and 'Size'. There is one entry, which is a file named 'hw1.java'. The file is highlighted with an orange oval and a callout pointing to it from the text 'Sub folders inside your folder'.

Name	Date modified	Type	Size
hw1.java	2/8/2016 1:31 PM	JAVA File	0 KB

In SML folder,

mchoorak_HW1 > SML				
<input type="checkbox"/>	Name	Date modified	Type	Size
	hw1.sml	2/8/2016 1:32 PM	SML File	0 KB

In Python folder,

mchoorak_HW1 > Python				
<input type="checkbox"/>	Name	Date modified	Type	Size
	hw1	2/8/2016 1:31 PM	Python File	0 KB

NOTE:

- Make sure all your programs upon execution must read from the input.txt file and write to output.txt file.
- You need **NOT** submit any input or output files
- Please **do not include testing code** in your submissions – this means your test program that contains main() for Java will not be submitted, also **NO function call to hw1()** in Python code or SML code. Our test script will have function calls to hw1().

Compress the UBITName_HW1 folder(UBITName_HW1.zip or UBITName_HW1.tar) and submit it on Timberlake using the command submit_cse305 your_file_name

Please note, late submissions will not be accepted.

For information on how to run the various programming language compilers/interpreters:

<https://wiki.cse.buffalo.edu/services/content/java>

<https://wiki.cse.buffalo.edu/services/content/standard-ml-new-jersey-smlnj>

<https://wiki.cse.buffalo.edu/services/content/python>

You will find resources for these languages on the Piazza course web site, under the ‘Resources’ page.

You might also find the following page, listing available CSE systems, helpful too:

<https://wiki.cse.buffalo.edu/services/content/student-systems>