CSE305: Assignment 1

February 12, 2017

Assigned: 2/13/2017 **Due:** 11:59pm 2/20/2017

Write a program in each of the following languages: **(60 points - 30 per language)**

1. Python **or** Java
2. SML

Your program will consist 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 is 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 the following to the *output.txt*:

```
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 for 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 on your local machine and verify you solution by using autolab. 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:

1. hw1.py
2. hw1.java
3. hw1.sml

Python Specific Instructions

Your code should be written in hw1.py. Create a *hw1* function that takes in two strings as arguments: the first for the name of input file and second for the name of output file.
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
```

Do not include this in your submission but use this line to test your code

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

Python Resources

Refer to 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.

SML Specific Instructions

Your code should be written in hw1.sml. Create a *hw1* function that takes in two strings as arguments: the first for the name of input file and second for the name of output file.
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, 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 result, true or false, to the output file *)
    (* Remember to flush the outStream, refer resources on TestIO. *)
```

Do not include this in your submission but use this line to test your code
val _ = hw1("input.txt", "output.txt")

SML Resources

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`.

Java Specific Instructions

Your code should be written in hw1.java. Create a *hw1* class with a static method *hw1* that takes in two strings as arguments: the first for the name of input file and second for the name of output file.

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

Skeleton Code:

```
Define hw1.java as:
    public class hw1 {
    public static void hw1(inFile, outFile){
        //function defintion 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*/
        hw1.hw1(input, output);
    }
}
```

If you are using Eclipse or any other IDE make sure you **do not create any packages** since this will not work with autolab. 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.)

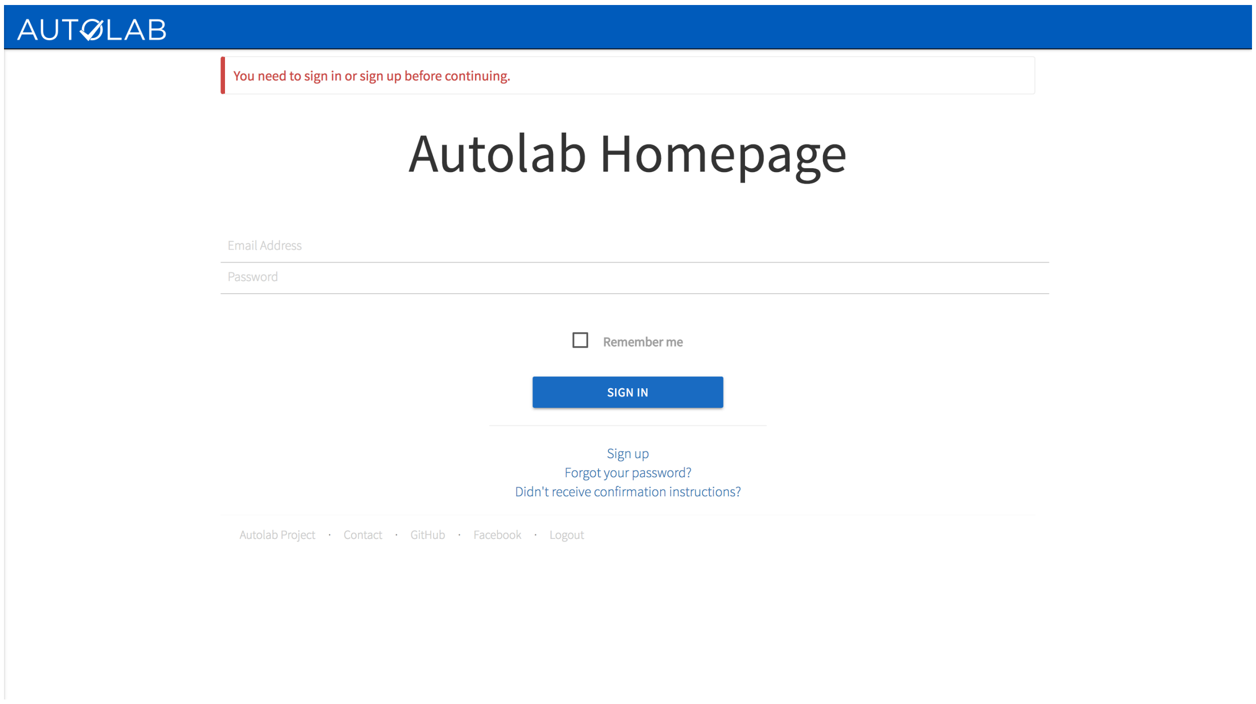

AUTOLAB
You need to sign in or sign up before continuing.
Autolab Homepage
Email Address
Password
Remember me
SIGN IN
Sign up
Forgot your password?
Didn't receive confirmation instructions?
Autolab Project · Contact · GitHub · Facebook · Logout

Figure 1: Autolab Home Screen

java Tester (without the .class extension)

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

Submission Instructions

Late submissions will not be accepted. You can use autolab to verify your program adheres to the specification outlined. Only your last autolab submission will be graded for you final grade. This means you can submit as many times as you want. If you have any questions please ask well before the due date.

Autolab account creation

An autolab account has been already created for you. You can access autolab at: `https://autograder.cse.buffalo.edu`. If you already have an account for autolab from a previous course or another course you are taking this semester you can log in normally and you should see CSE305 in your list of courses.

If you have not used autolab before you will need to go to: `https://autograder.cse.buffalo.edu/auth/users/sign_in` and click on the *Forgot your password?* link. The following page will prompt you for your email address, please you use buffalo email address with your UBIT id as this is the email that was used to create your account for you. You will receive an email with instructions on how to reset your password and log into autolab. It is important you verify your account is working well in advance of the turn in date.

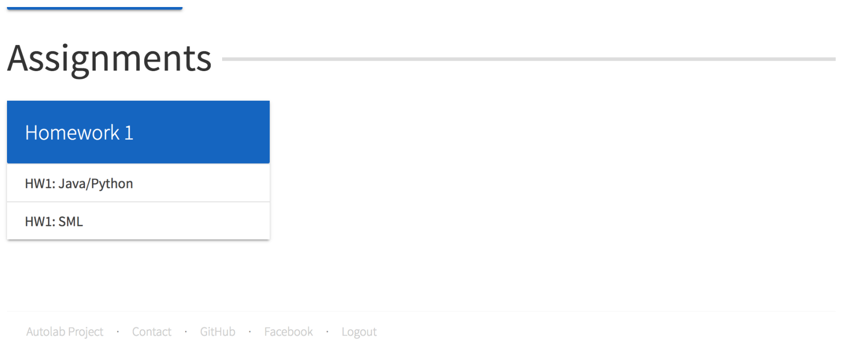


Figure 2: CSE305 Assignment 1 Autolab Overview

Autolab submission instructions

When you log into autolab and pick the CSE305 course you will see an assignment called HW1. HW1 has two separate parts – one for SML and one for Java/Python. You will need to submit a solution to each of these parts to receive full credit for the assignment. Each part is worth 30 points and the total for HW1 is 60 points.

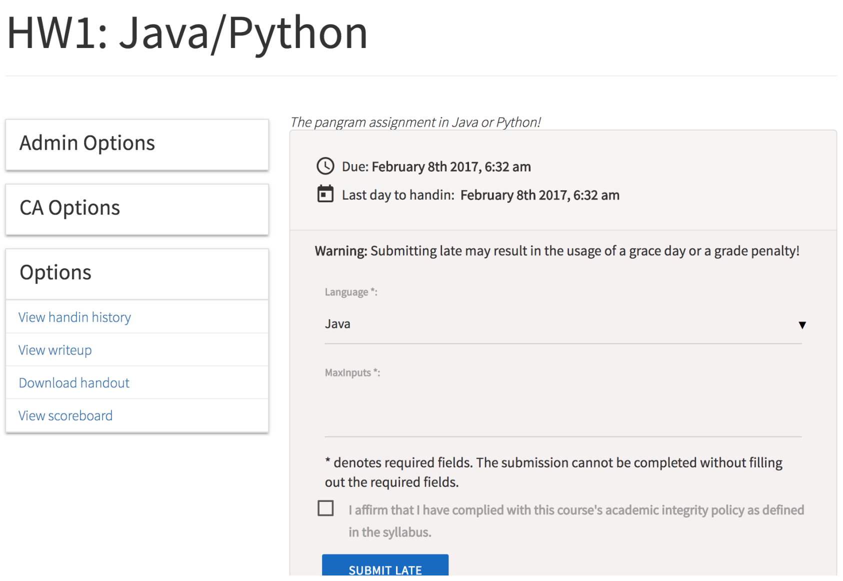
On the page for each individual part of HW1, select the language you used from the drop-down menu (not applicable for SML), and set the number of inputs you want your code to be tested on. Click *Submit* and choose your program. Your program can be in a zip or tar archive, or just the .py/.sml/.java file if the program is self-contained. If you are submitting an archive, make sure the 'hw1' file is NOT in any folder. Your program will not be found by the grader and you will receive no credit. Of course, you can submit as many times as you wish before the deadline.

Additional Resources

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`



HW1: Java/Python

The pangram assignment in Java or Python!

Due: February 8th 2017, 6:32 am

Last day to handin: February 8th 2017, 6:32 am

Warning: Submitting late may result in the usage of a grace day or a grade penalty!

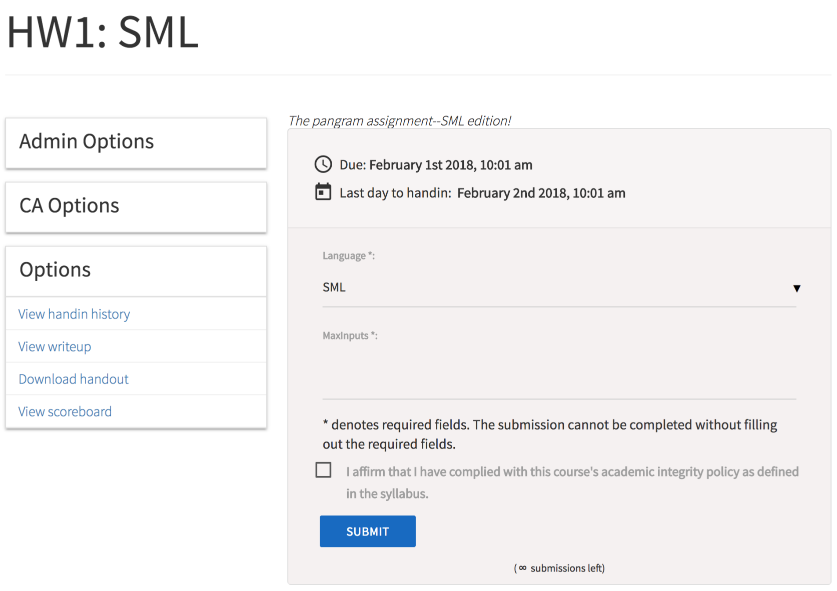
Language *: Java

MaxInputs *:

* denotes required fields. The submission cannot be completed without filling out the required fields.

I affirm that I have complied with this course's academic integrity policy as defined in the syllabus.

Figure 3: Java / Python Submission



HW1: SML

The pangram assignment--SML edition!

Due: February 1st 2018, 10:01 am

Last day to handin: February 2nd 2018, 10:01 am

Language *: SML

MaxInputs *:

* denotes required fields. The submission cannot be completed without filling out the required fields.

I affirm that I have complied with this course's academic integrity policy as defined in the syllabus.

(∞ submissions left)

Figure 4: SML Submission