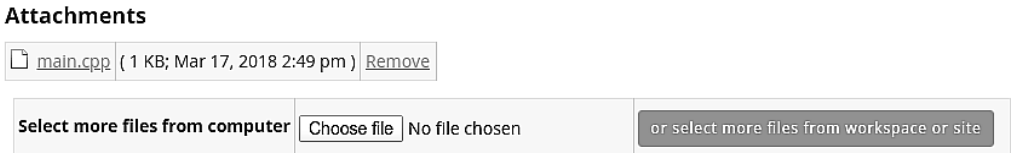
School of Electrical and Information Engineering
University of the Witwatersrand, Johannesburg
ELEN2004/ELEN2020 – Software Development I

Practical Programming Test 2

4 June 2020

Instructions

- You have 90 minutes to complete and submit your solutions (including online submission).
- Remember that even though you are writing this test remotely you should spend time reading and understanding the task you must complete as well as planning the code on paper before diving into your programmed solution.
- Failure to submit before the deadline will be treated as a failed practical programming test and will be awarded a score of 0. Keep an eye on the time.
- Your **submission** will consist **only** of your source code files i.e.
  - `yourFileName.cpp`
- Submissions are made using Sakai. The submission link is found under **Assignments**. Your submission should appear as follows on Sakai



- Make sure that you receive the submission confirmation email.
- Your solution will be assessed on
  - coding style,
  - simplicity,
  - use of comments,
  - clarity,
  - ability to compile, run and generate correct output.

  The solution must comply exactly with the specifications and requirements given in the problem definition.
- **Important:** You must make sure you behave in a professional and ethical manner.

Task

An anagram is a word or phrase formed by rearranging the letters of a different word or phrase, typically using all the original letters exactly once. For example, the word **anagram** can be rearranged into **nag a ram**, or the word **binary** into **brainy**. You are required to read in an unknown number of line pairs (containing words/phrases) and determine if they are anagrams of each other.

The input file, named `input.txt`, will contain an even number of lines with pairs of phrases/words to compare. The phrases or words to be compared are separated by a newline character. Different pairs will also be separated by a newline character.

For example the input file may contain the following:

Listing 1: `input.txt`

```
binary
brainy
anagram
nag a ram
```

Here we have two pairs of phrases to check (i.e. 4 lines). The first pair consists of **binary** and **brainy**. We can see that rearranging the letters of the word **binary** we can obtain the word **brainy**. The second pair consists of **anagram** and **nag a ram**. The second pair are also anagrams of one another as we can rearrange the order of letters of **anagram** and ignoring spaces, get to the phrase **nag a ram**.

You will be required to write the result to an output file named `output.txt`. If a pair of words/phrases are anagrams a 1 should appear in the output file. If a pair of words/phrases are not anagrams a 0 should appear in the output file.

Listing 2: `output.txt`

```
1
1
```

The input file which you will read the words/phrases from is named `input.txt`.

Note the following:

- Each line in the input file may contain any number of words separated by spaces and/or punctuation.
- The input file may contain any number of lines with the number of lines always being even (as pairs of words/phrases will appear across two lines).
- All punctuation should be ignored.
- Upper case and lower case letters are regarded as the same i.e. 'a' and 'A'.
- The result of whether each pair is an anagram or not must be written to a file named `output.txt`. A 1 should be written for pairs which are anagrams and a 0 for pairs which are not anagrams.
- Your output file should therefore contain half the number of lines which your input contains.

In addition to the above take note of the following:

- **You may not make use of third party libraries such as `<algorithms>` or any built in sort functions.**
- **You may make use of vectors, arrays and strings.**

- **Your code must contain at least 1 user defined function.**

Useful functions include:

- `ispunct(ch)` which accepts a `char` and returns 1 if `ch` is a punctuation mark and a non-zero integer if it is not.
- `isspace(ch)` which accepts a `char` and returns 1 if `ch` is a space and a non-zero integer if it is not.
- `str.erase(index,l)` returns the `string`, `str`, with the characters starting at position `index` for length `l` removed.

Example 1

Note that each element of a pair is separated by a newline character. Pairs are also separated by a newline character.

Listing 3: `input.txt`

```
Tar
Rat
red car
cedar tree
school Master
the classroom
listen
Silent
learning
studying
```

Using the input file `input.txt` given, your code would need to compare the following phrases/-words:

- **Tar** and **Rat** contain the same letters so they **are** anagrams.
- **red car** and **cedar tree** do not contain the same letters so they **are not** anagrams.
- **school Master** and **the classroom** contain the same letters so they **are** anagrams.
- **listen** and **Silent** contain the same letters so they **are** anagrams.
- **learning** and **studying** do not contain the same letters so they **are not** anagrams.

Therefore the corresponding output file, `output.txt`, should contain the following:

Listing 4: `output.txt`

```
1
0
1
1
0
```

Example 2

Listing 5: `input.txt`

```
the Morse Code?
here comes dots!
Slot machines
cash lost in me
paris
pairs
socks
shoes
```

Listing 6: `output.txt`

```
1
1
1
0
```