COSC 1P03 Assignment 1

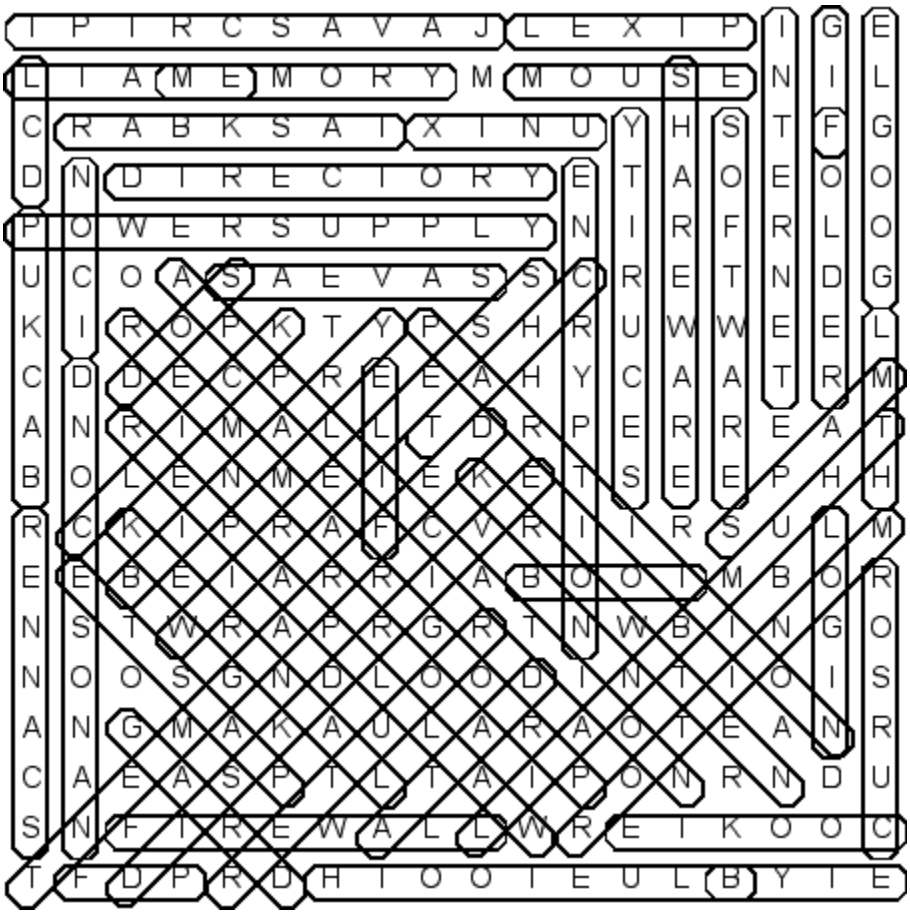
"Puzzling it out"

***Due: Feb. 3, 2017 @ 4:00 pm** (late date Feb. 6 @ 4:00 pm)*

In preparation for this assignment, create a folder called `Assign_1` for the DrJava project for the assignment. The objective of this assignment is to apply arrays in the development of a problem solution.

WordSearch Puzzle

A wordsearch puzzle is a square or rectangular grid of letters. Within the grid are a number of words that may be written right, left, up, down and diagonally up-right, up-left, down-right and down-left. The goal is to find all the words within the grid. Usually a list of words is supplied. A sample wordsearch puzzle (and its solution) is shown below:

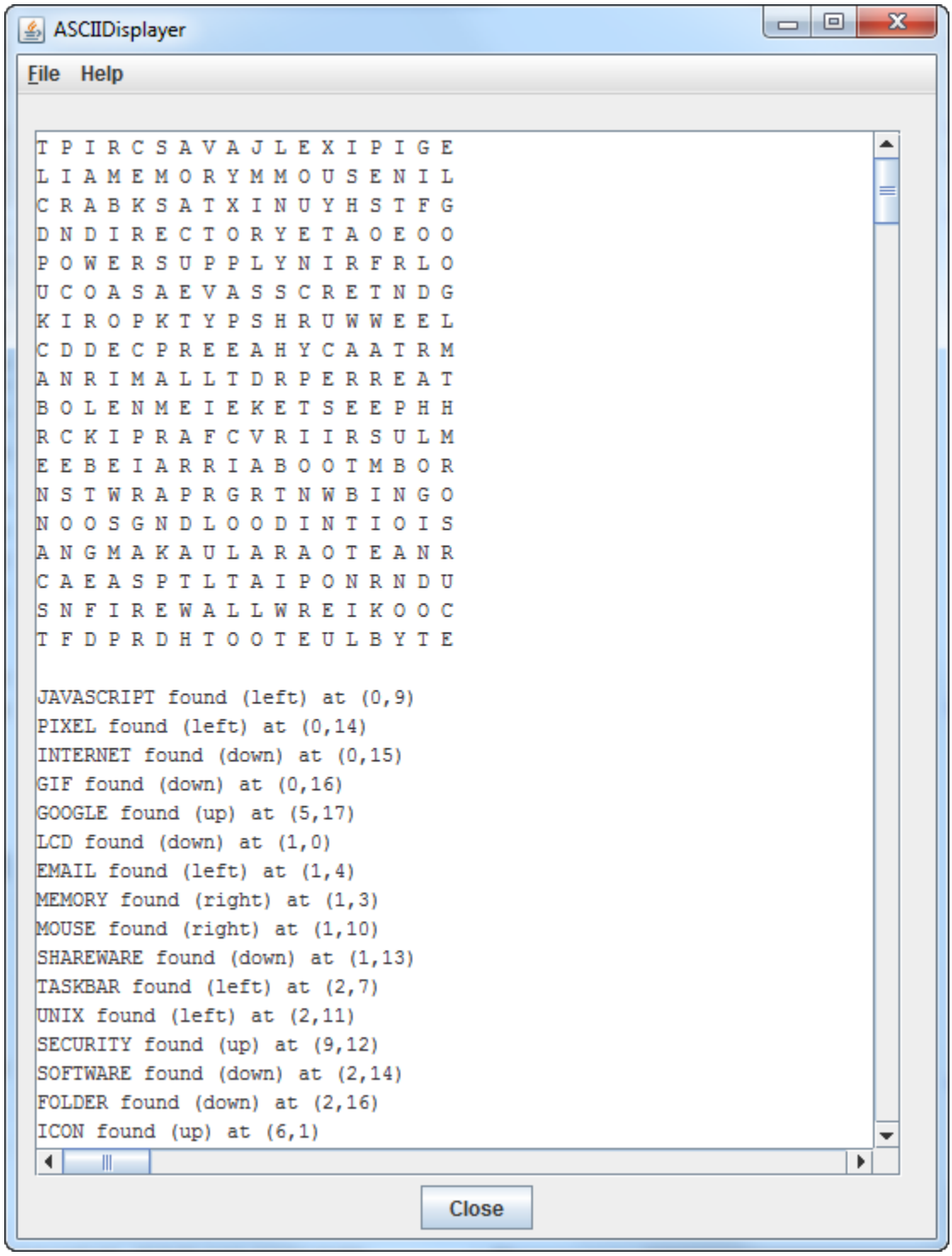


For this assignment you will develop a solver for such puzzles.

The puzzle is prepared as an `ASCIIDataFile`. On the first line are two integers giving the height and width of the puzzle. Following that are *height* lines consisting of *width* characters (`char`), tab delimited, making up the puzzle. Finally there are some number of lines each containing a word (`String`), being the words that are to be located within the puzzle.

The output is written to an `ASCIIDisplayer` with text area: 35×70. The puzzle should first be displayed, followed by one line per word, indicating where the word as found in

the puzzle. For example, the first part of the output for the above puzzle would look like:



Hints

- The puzzle can be represented as a right-sized two-dimensional array of characters (`char`).
- A `String` can be converted into a right-sized array of characters via the `String` method `toCharArray`.

- A word can occur in any of 8 directions starting from a square in the puzzle:
  - to the right (normal text: `MEMORY` above)
  - to the left (backwards: `PIXEL`)
  - downwards (`INTERNET`)
  - upwards (`GOOGLE`)
  - diagonally downwards to the right (`APPLICATION`)
  - diagonally downwards to the left (`CHAT`)
  - diagonally upwards to the right (`WIRELESS`)
  - diagonally upwards to the left (`PROGRAMMER`)
- Consider writing separate methods such as:

```
private boolean searchRight ( int x, int y, String word ) {
```

  that look for a `word` starting from position `(x,y)` in the puzzle and proceeding in the designated direction (right in this case). The method returns `true` if the word is found in the designated direction starting from the indicated position. You would have 8 such methods. Each of these methods will be quite similar.

- A word cannot occur starting from `(x,y)` if there isn't enough room from `(x,y)` to the edge of the puzzle in the designated direction. For example, if the puzzle is 10×10, we are checking from position `(3,7)` to the right and the word is more than 3 characters in length.
- Consider developing the solution in phases such as:
  1. read and display puzzle
  2. check for a single word in one direction only (e.g. right)
  3. repeatedly add the ability to check for a single word in each of the other 7 directions
  4. add the ability to search for multiple words.
- Don't use the big puzzle for testing. Make special small puzzles to test each of the 8 search methods above such as the included puzzle `right.txt` that has only one word written to the right.

Submission:

Details regarding preparation and submission of assignments in COSC 1P03 are found on the COSC 1P03 Sakai Site as `Assignment Guidelines` under `Course Documents`. This document includes a discussion of assignment preparation, programming standards, evaluation criteria and academic conduct (including styles for citation) in addition to the detailed assignment submission process copied below.

To prepare and submit the assignment electronically, follow the procedure below:

1. Ensure your folder (say `Assign_1`) for the assignment accessible on your computer..
2. Using DrJava, print (to `CutePDF Writer`) each of the `.java` files of your assignment using the name *`ClassName`*`.pdf` where *`ClassName`* is the class name (i.e. same name as the `.java` file) and save the `.pdf` file at the **top level** of the project folder (i.e. directly within `Assign_1`).
3. Run the program using the data file `COSC 1P03 PUZZLE.txt`. When the program is finished but before you press `Close` on the display, select `File/Print Image of Window`… and print to `CutePDF Writer` as

`puzzleSolution.pdf` at the **top level** of the project folder (i.e. directly within `Assign_1`).

4. The submission folder should now include the `.java`, `.class` (created by DrJava) and `.pdf` files for the class you wrote. It should also include the `.pdf` output file created when running the program (above).
5. Create a `.zip` file of your submission by right-clicking on the top level folder (i.e. `Assign_1`) and selecting `Send to/Compressed (zipped) folder`. A zipped version of the folder will be created. Use the default name (`Assign_1.zip`).
6. Log on to Sakai and select the COSC 1P03 site.
7. On the `Assignments` page select `Assignment 1`. Attach your `.zip` file (e.g. `Assign_1.zip`) to the assignment submission (use the `Add Attachments` button and select `Browse`). Navigate to where you stored your assignment and select the `.zip` file (e.g. `Assign_1.zip`). The file will be added to your submission. Be sure to check the `Honor Pledge` checkbox. Press `Submit` to submit the assignment. You should receive a confirmation email.
8. Assignments incorrectly submitted will lose marks. Assignments without the required files may not be marked.

DrJava

The `.zip` folder you submit should contain the project folder including all files relevant to the project—the `.drjava`, `.java` and `.class` files for the assignment and `.pdf` files for program listings and output.

Other Platforms

If you are using an IDE other than DrJava to prepare your assignment, you must include the `.java` source files and the `.pdf` files described above as well as a file (likely `.class` or `.jar`) that will execute on the lab machines.