AUB

Department of Computer Science

CMPS 200 – Introduction to Programming

Exam 2

1. Index.

At the end of books, there is usually an index that lists the pages where a certain word appears. In this problem, you will create an index for a text but, instead of page number, you will use the line numbers. You will write a program `index.py` that reads the name of a text file and a sequence of words. For every word in the list, your program will find the lines in the text file where the word occurs and print the corresponding line numbers where the numbering starts at 1. Your program should open and read the file only once. A word may appear more than once on a line, but should be indexed only once per line. The sample input file `onteaching.txt` in the sample invocation below can be downloaded from Moodle.

```
> python index.py onteaching.txt wisdom knowledge understanding science
wisdom  5, 7
knowledge  2, 22, 23
understanding  10, 11, 24
science  17
```

2. Squares.

- Write a function `square(x, y, d)` that draws a square of side length d centered at (x, y).
  Hint. With the pen up, go to the lower left corner, put the pen down and move/turn four times.
- Write a recursive function `squares(x, y, d, n)` that draws a pattern of squares centered at (x, y). When $n = 0$ nothing is drawn. When $n = 1$ a single square of side length d is drawn. When $n = 2$, the pattern consists of a main square of side length d and four smaller ones, each centered at a corner with a side length equal to $\frac{d}{2}$. The pattern is repeated recursively for larger values of n. The patterns drawn for $n = 1$, 2, and 3 are shown below.

3. Transpose.

Consider a 2D $n \times n$ table of integers represented as a list of lists. There are n lists representing the rows of the table, and each row has n integers.

- Write a function `build_table(lst)` that takes in a list of n^2 integers, and builds and returns a 2D list representing an $n \times n$ table. The elements of the list are arranged in row order: elements of the first row are first, followed by the elements of the second row, etc. Your function should include an `assert` statement to make sure that the number of elements in the input list is a perfect square (e.g., 1, 4, 9, 16, 25, etc.)
- Write a function `transpose(table)` that builds and returns a new $n \times n$ table whose rows are the columns of the original table. The function should leave the object passed to it intact.
- Write a function `transpose_inplace(table)` that does not create a new table, but instead modifies the table object passed to it.
- Write a program that reads n^2 integers from the command line, builds a table, and prints its transpose.

```
> python transpose.py 1 2 3 4 5 6 7 8 9
1 4 7
2 5 8
3 6 9
```