CS2104 – Programming Languages
School of Computing
National University of Singapore

September 7, 2012

Problem Set 3

Semester 1, 2012/13

Due: ***September 23, 23:59***

15 marks

Submission: In IVLE, in the cs2104 workbin, you will find a folder called `“Homework submissions”`. In that folder, there are currently ***3 subfolders:*** **`PS3P01`**, **`PS3P02`**, and **`PS3P03`**. The last two digits of the folder name indicate the solution that is to be submitted into that folder: the solution to *Question 1* into **`PS3P01`**, and so on (that is, you need to submit 3 separate solutions to 3 problems). A solution should consist of a ***single text file*** that can be compiled, or loaded into the interpreter of interest and executed. You should provide as much supplementary information about your solution as you can, ***in the form of program comments***. Moreover, if you work in a team, state the members of the team at the beginning of the file, in a comment. You do not need to submit the same file twice, one submission per team is sufficient.

To solve the first 3 problems of this problem set, you will need to install the image processing package *ImageMagick*, available at `www.imagemagick.org`. (Linux users may get it by using the software update utility). This package provides the command `convert`, that can be used from the command line prompt. The command has a complicated language of options, and is capable to performing all the standard image processing operations. However, you only need to be familiarized with the following 3 operations:

```
convert –rotate 90 input.jpg output.jpg

convert –scale 50%x25% input.jpg output.jpg

convert +append input1.jpg input2.jpg output.jpg
```

The first command rotates the `input.jpg` image 90º clockwise, and writes the result into the `output.jpg` image. The second command scales the `input.jpg` image 50% on the horizontal direction, and 25% on the vertical direction, and writes the result into `output.jpg` (obviously, the percentages can be any values between 0-100). The third command collates `input1.jpg` and `input2.jpg` along the horizontal direction, and places the result in file `output.jpg`.

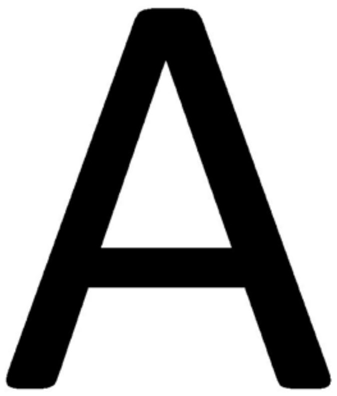
You are encouraged to test the commands above on the sample images `a.jpg` and `b.jpg` provided with the problem set.

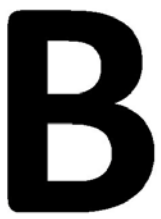
The theme of this problem set is compiling a graphic description language into sequences of image processing commands. The image processing commands can only be from among the three listed above. **The only parts that you will be allowed to change in these commands are:**

- The names of input and output files
- The scaling percentages in the second command.

You will not be allowed to change the angle of rotation, nor will you be allowed to use other options of the `convert` command.

As an example of using these commands, consider the following scenario. Assume that the images `a.jpg` and `b.jpg` are the following:

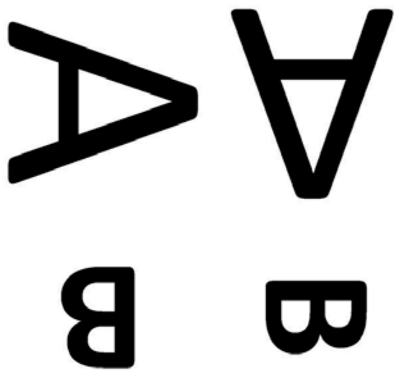




Consider now the following sequence of commands passed to the shell (either `bash`, or `cmd.exe`):

```
convert -scale 50%%x50%% a.jpg o111.jpg
convert -scale 50%%x50%% b.jpg o1121.jpg
convert -rotate 90 o1121.jpg o112.jpg
convert +append o111.jpg o112.jpg o11.jpg
convert -rotate 90 o11.jpg o1.jpg
convert -scale 50%%x50%% a.jpg o2111.jpg
convert -rotate 90 o2111.jpg o211.jpg
convert -scale 50%%x50%% b.jpg o212.jpg
convert +append o211.jpg o212.jpg o21.jpg
convert -rotate 90 o21.jpg o2.jpg
convert +append o1.jpg o2.jpg o.jpg
```

The final image, `o.jpg`, looks like this:



Problem 1 [4 marks, submit to `PS3P01`] [A]

We define the following graphics description language. An `Image` is either:

- A Prolog atom, representing the `.jpg` image file with the same name (i.e. the atom a represents the file `a.jpg`, **assumed to be of size `640x640`**)
- The Prolog term `beside(Image1,Image2)` represents the image obtained by collating the images resulting from the evaluation of the terms `Image1` and `Image2` along the horizontal axis (`Image1` should be placed to the left of `Image2`). Both images should be shrunk horizontally to half their size, so that the result has the same size as each of the original images.
- The Prolog term `rotate(Image)` represents the image obtained by rotating 90º clockwise the image resulting from the evaluation of the term Image.

All the resulting images are expected to be of size 640x640, the same as the size of the original images.

For instance, the term

```
beside(rotate(beside(a,rotate(b))),rotate(beside(rotate(a),b)))
```

represents the `o.jpg` image presented above.

Write a predicate called `montage` that takes two arguments: a graphic description expression, and an output file name. The predicate should compile the graphic description expression given in its first argument into a sequence of `convert` commands which, when run at the command prompt, generate the described image. The generated commands should place the result image in the file whose name is given as the second argument to the predicate.

For instance, the following query:

```
?- Prog = beside(rotate(beside(a,rotate(b))),rotate(beside(rotate(a),b))),
   montage(Prog,o).
```

should generate the sequence of commands given above, in the introductory part of the problem set (remember, the atom `o` stands for the file `o.jpg`). Note that the predicate should simply print the commands on the screen, and not be responsible for running them. The user is expected to copy and paste the commands into the shell window in order to produce the output. The temporary file names used in the process of generating the final output are not a compulsory requirement of this problem. You should feel free to come up with your own naming scheme, as long as the final image is placed in the file with the specified name.

Problem 2 [1 marks, submit to PS3P02] [A]

We expand the language we defined in Problem 1 with assignment. Our new programs have the format:

```
filename1 = Expr1 ; filename2 = Expr2 ; … ; filename = ExprN
```

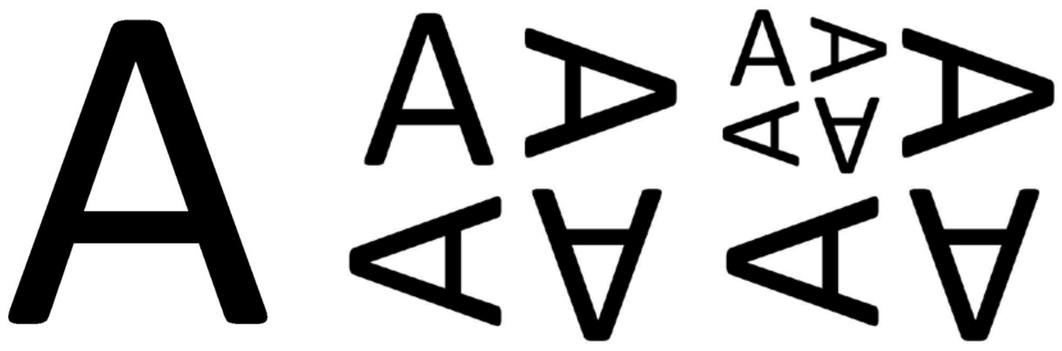
The semantics of these programs is the following. The result of evaluating `Expr1` should be placed in filename `filename1.jpg`. Now, `Expr2` may reference `filename1`, and, after its evaluation, the result should be placed in `filename2`. This continues sequentially till all the assignments in the program are exhausted. Every new expression may reference all the filenames previously defined. For instance, the expression

```
x=rotate(beside(a,rotate(b)));y=rotate(beside(rotate(a),b));o=beside(x,y)
```

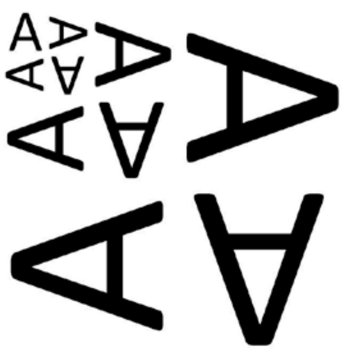
should evaluate to the same `o.jpg` image as before. Write a predicate `ma` (as in Montage with Assignment), that takes a program of the format described above as argument, and writes out the `convert` commands that generate each file on the left side of an assignment.

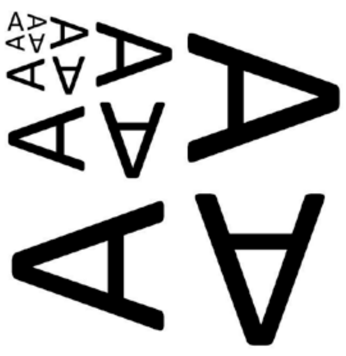
Problem 3 [4 marks, submit to PS3P03] [A]

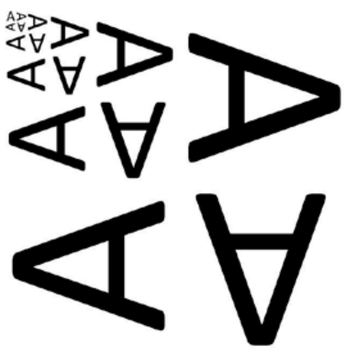
Consider the following set of images:



n=0 n=1 n=2


n=3


n=4


n=5

Write a predicate called `tile`, which takes 3 arguments. The first argument is the level of the image, represented by the value of `n` underneath each picture in the set given above. The second is the input image, which should be `a` for the set above. The third argument is the name of the output image. Thus, the 6 images given above should be generated by the output of the following queries:

```
?- tile(0,a,output0).
?- tile(1,a,output1).
?- tile(2,a,output2).
?- tile(3,a,output3).
?- tile(4,a,output4).
?- tile(5,a,output5).
```

Problem 4 [2 marks, submit to PS3P04] [C]

Consider a binary encoding of natural numbers as reversed lists of bits. Thus, the list `[1,1,0,1]` stands for the binary representation 1011, equal to 11 (eleven) in base 10. Write a Prolog predicate `add` that implements the addition of numbers represented in this way. The predicate should allow for the following interaction:

```
?- add([1,0,1],[1,1],X). % 5+3=8

X = [0,0,0,1]
```

Problem 5 **[2 marks, submit to PS3P05] [C]**

For the binary encoding described in Problem 4, write a multiplication predicate that works in a similar manner. Your solution should reuse the predicate developed in the previous exercise.

Sample interaction:

```
?- mul([1,0,1],[1,1],X). % 5*3=15

X = [1,1,1,1]
```

Problem 6 **[2 marks, submit to PS3P06] [C]**

Declare the operator `$=` that evaluates expressions made up of binary lists as operators, and the multiplication and addition operators `*` and `+`.

Sample interactions:

```
?- X $= [1,1]+[0,1]*[0,1]. % 7 = 3+2*2

X = [1,1,1]

?- X $= ([1,1]+[0,1])*[0,1]. % 10 = (3+2)*2

X = [0,1,0,1]
```

Hint: Reuse the predicates defined in the previous two problems. Perform an appropriate `op` declaration for the `$=` predicate; this declaration should be similar to the one for the operator `=` .