CP2 Exercise 8 **11/12/2006**

Prerequisites:
First install a local copy of the C-Sheep system on your partition:

- go to the directory ~eanderson/C-Sheep : > **`cd ~eanderson/C-Sheep`**
- run the install makefile: **`> make install`**

This will create 4 local directories (*include, lib, bin and meadow*) and add some additional paths to your .bashrc file.
You will need to resource .bashrc or alternatively open a new terminal.
"The Meadow" is stored in the meadow directory. You can run it with the default sheep program just by calling the **`meadow`** program or run a specific C-Sheep program by using **`meadow -program progname.csx`** .
C-Sheep programs can be compiled for the meadow by calling scc:

```
scc progname.c -o progname.csx
```

C-Sheep programs can be compiled using gcc:

```
gcc -ansi progname.c -lSDL -lGLU -lGL -o progname
```

1. "Haggis" the sheep has a problem: He is at the entrance of a maze (43/11). His favourite ball however is at an exit on the opposite end of the maze.
Re-implement the demonstration from the last lecture by:

- developing an algorithm that uses the "follow the right hedge/wall" strategy
- deriving a C-Sheep program from that algorithm
- testing that program either using "The Meadow" or the companion library

Then modify the program from an iterative to a recursive solution:

- use a function "go" which attempts to take the next step for the sheep and which calls itself (*recursively*) until the ball is found.

Use makefiles.

2. Expand the above program so that "Haggis" the sheep returns to the entrance of the maze after he finds his ball. This should be done as an addition to the recursive function. Use makefiles.

You should not spend more than one hour and 15 minutes on solving the above 2 exercises!

3. Look again at last week's tree library exercise (*exercise 2*). Expand your library by first designing the algorithms and then implementing respective functions for saving and reloading the binary tree in a file.
You should take into consideration:

- what would be the requirements in regard to datatypes used and functions employed, necessary for saving a binary tree to a file?
- how would the algorithm that saves the binary tree out into a file work if it should also be able to reload that tree into memory, keeping the original tree structure intact?
- how would the algorithm for (*re*-)loading the tree into memory work?

Expand your original tree program to use these functions. Use makefiles.

Note: *look at the tree traversal algorithms – some might be useful. Also you might need an intermediate datatype for holding node data to solve this as it would be unsafe to try to store addresses.*

The following is a sample makefile:

```
# sample makefile

OBJECTS = file1.o file2.o
PROGNAME = program
OUTPT = -o
OPTIONS = -ansi -c
COMPILER = gcc

# build targets

$(PROGNAME): $(OBJECTS)
    $(COMPILER) $(OBJECTS) $(OUTPT) $(PROGNAME)

# generation of file1
file1.o: file1.c
    $(COMPILER) $(OPTIONS) file1.c

# generation of file2
file2.o: file2.c
    $(COMPILER) $(OPTIONS) file2.c

clean:
     @ echo "remove build targets"
     @ rm $(OBJECTS)
     @ $(PROGNAME)
```