Decimal to Binary Conversion
CMPS 101 Program 1, Spring 2014

Assigned: 03-31-14 Due: Friday 04-04-14

1 Introduction

This assignment is to convert decimal numbers to their binary representation. It is intended as a warm-up to ensure everyone is familiar with C, ADTs, separate compilation, make files, UNIX utilities, and the submission/grading system. The main goals of this assignment are:

- A warmup in C. For those less familiar with ANSI C, this will give you chance to get some practice with it.
- To use a pre-defined and coded ADT written by others. This will this give you a better idea of how ADTs work, and the value of comments.

Key Skills: ANSII C programming and debugging, input/output, multi-file compilation, ADT use, makefiles, zip, E-commons submission

2 Assignment Description

First, please read the ADT handout posted on E-commons for more information on ADTs, header files, and separate compilation.
Second, if you are not an expert on Makefiles, read the makeTutorial posted on E-commons.
Third, read the C-style handout on E-commons.
Fourth, download the intstack files from E-commons. These files include:

1. `myinclude.h` A set of libraries and definitions I commonly use in my programs. Note that this `.h` file includes other `.h` files contrary to the advice in the ADT handout. However, I tend to forget to include these standard libraries in my C programs, and `myinclude.h` serves as a reminder to me. Feel free to edit the file to remove these inclusions and place them in the appropriate `.c` files.

2. `intstack.h` The header file for the intstack module. Ideally the comments in this file are all you need to determine what the intstack functions do and how to use them.

3. `intstackdr.c` A program designed to "test drive" the intstack module and verify that it is working correctly in at least some cases. This also serves as an example of how the intstack module should be used.

4. `intstack.c` The implementation of the intstack module – you need this to compile the module. Ideally you should never need to look at it, but sometimes examining the actual code for a function can give you insights the comments do not.

5. `Makefile` To build the intstack module and an `intstackdr` executable for the driver.

6. `README` A file that lists the files associated with the intstack ADT and their purposes.

You should first compile the module (using the Makefile) and run the driver program. Scan the driver program to ensure that the module seems to be working properly. Modifying the driver program can be a very productive way of playing with the ADT to see how it behaves.

Then you should design and write a simple self-documenting ANSII C program that reads in a single non-negative decimal integer from standard input and implements the following algorithm using the intstack module to convert the number to its binary representation.

```
create an empty stack
if the number is 0, then push a 0 onto the stack
while the number is positive,
    if the number is odd, push a 1 onto the stack
    else   push a 0 onto the stack
    set the number to the number divided by 2 (rounded down)
end while
while the stack is not empty
    print the top number on the stack,
    and pop it off the stack
end while
```

Ideally, the program would output a meaningful message and terminate if the input is not a non-negative integer.

Important notes:

- Call your program `decToBinary.c`.
- Create a Makefile (or modify the intstack one) to do all needed compilations to build the intstack module and a `decToBinary` executable.
- Create a README file (or modify the instack one).
- *Do not change* the `intstack.h` and `instack.c` files, and do not implement your own stack or use another method to do the decimal to binary conversion.

Before submitting your program, ensure that the makefile and program both work on the campus unix.ic system. Make sure that all of files you created/modified have header comments indicating the authors/editors and acknowledgements of any help received.

3 Submission

To submit your assignment, create a `.zip` file called `LastNameFirstInitialProg1.zip` (where `LastName` is your last name, and `FirstInitial` is your first initial, thus I would submit `HelmboldDProg1.zip`) containing the seven files: `intstack.c intstack.h intstackdr.c decToBinary.c myinclude.h Makefile README`, and turn in the zip file via the prog1 assignment in E-commons.

4 Grading

The assignment is worth 5 points.

- 2 points for correctness, including using the intstack ADT as described.
- 1 point for a good makefile
- 1 point for correct submission of all files
- 1 point for good style, *all* of: code structure, good identifiers, header comments, and README file.

5 Example Execution

A correct program will produce a dialog like the following:

```
Input a non-negative decimal integer to convert to binary:
> 1234
Decimal 1234 is 10011010010 in binary.
```