Mechanics Ahnentafel

CSCI-243 Homework

5/14/2018

1 Background

Binary numbers and conversion between binary and decimal are topics that come up in many contexts related to computing. As you are probably aware, binary is a number system that uses 0s and 1s to represent each number instead of the ten digits (0,1,2,3,4,5,6,7,8,9) that we use with decimal representation.

One advantage of this is that since we only need to represent two digits, this can be simulated with in electronic circuits more easily.

In this project, you will be working with binary values and converting them to decimal and also converting from decimal to binary. If you have never done this conversion or would like to brush up on the concept you should look here and here.

2 Introduction

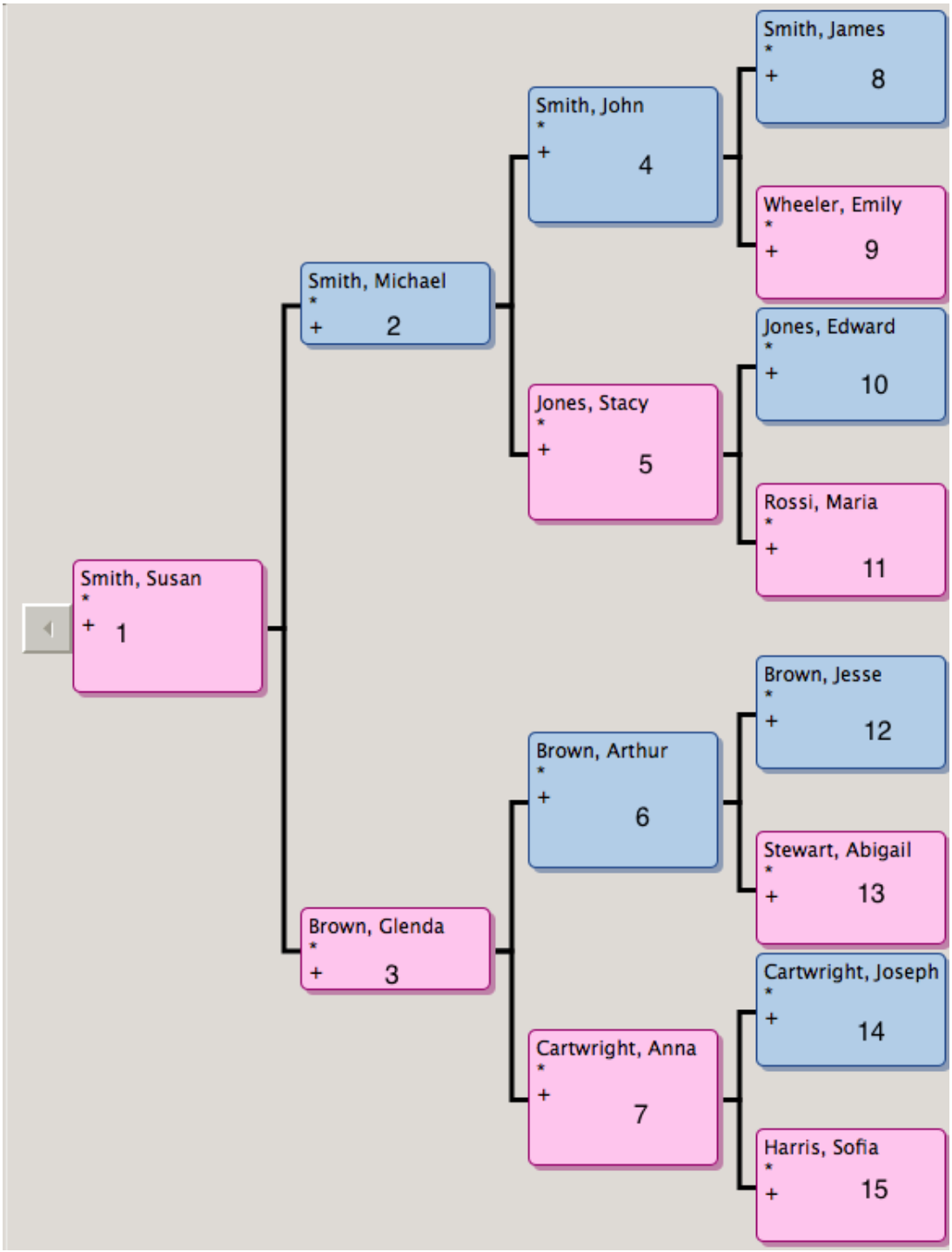
For this project, you will be implementing a menu-based program that allows for conversion of Ahnentafel numbers between binary and base-10 and also a written description of the number.

Ahnentafel numbers are used in genealogy to determine the relationship between the base individual and one of his/her ancestors. It is convenient to show familial relationships in this manner since each number unambiguously represents a specific ancestor, and the degree of relation to the target individual can be determined rather easily.

For example, starting with an individual and listing their own ahnentafel number and continuing up through both sets of grandparents can be represented as follows:

Decimal	Binary	Relation
1	1	self
2	10	father
3	11	mother
4	100	father's father
5	101	father's mother
6	110	mother's father
7	111	mother's mother

Here is a small sample family tree. On this tree, you can see the ahnentafel numbers for the the individuals up to 15. For example, Maria Rossi is number 11 which is 1011 in binary and corresponds to the individual's father's mother's mother. This is with respect to Susan Smith our starting individual.



You may have noticed a pattern in the last example. When the number is represented in binary, each time you see a "1", that corresponds to mother, and each time you encounter a "0", that is the father (with the exception of the starting individual). Using this technique, it is possible to represent any individual in the ancestor chart in a unique and unambiguous way.

Take another example of Emily Wheeler. This individual's ahnentafel number is 9 which is 1001 in binary. So, starting with a "1" for the individual and evaluating this left-to-right,

you get the individual's father's father's mother. This corresponds exactly to their position in the ancestor chart (again, relative to Susan Smith).

Another property of representing ancestors this way is that you can always find the ahnentafel number of a given individual (not only the starting individual) by taking their number and doubling it for their father (2x), or doubling it and adding one for their mother (2x+1). For example, look at Glenda Brown. Her ahnentafel number is 3. Her father's is 2x3 = 6 and her mother's is 2x3+1 = 7. You can use a similar approach to find grandparents, great-grandparents and so on.

For additional information and examples see: Ahnentafel.

3 Requirements

3.1 How the Program Should Work

Your program will have both a menu-based and command line options. When run without any command line parameters, your program should display the following menu:

```
1) description
2) ahnentafel number (base 10)
3) ahnentafel number (base 2)
4) relation (e.g. mother's mother's father)
5) exit

>
```

- Item '1' in the menu is simply a description of ahnentafel numbers (see below for output).
- Item '2' allows the user to enter the ahnentafel number in base 10, and will perform the conversion to base 2 and print that result as well as an English description of the connection from the base individual to the target ancestor. Finally, the number of generations back will be calculated and displayed. (See How to calculate the number of generations back, below).
- Item '3' allows the user the enter the ahnentafel number in binary (base 2), and will convert to base 10, print out the result and also print out the English description. The number of generations back will also be calculated and displayed.
- Item '4' allows the user to enter the relation and will print out the ahnentafel number in binary, the ahnentafel number in base 10 and the number of generations back the individual is from the starting individual. See details below.
- Item '5' will print "Goodbye." and exit the program with `EXIT_SUCCESS`.

3.2 Example Program Runs

Your program should look like the following example (run without command line parameters):

(Note: in these examples the user enters data following the prompt. You could redirect input such that it is coming from a file.)

```
$ ahnentafel

1) description
2) ahnentafel number (base 10)
3) ahnentafel number (base 2)
4) relation (e.g. mother's mother's father)
5) exit

> 1
The Ahnentafel number is used to determine the relation
between an individual and each of his/her direct ancestors.

1) description
2) ahnentafel number (base 10)
3) ahnentafel number (base 2)
4) relation (e.g. mother's mother's father)
5) exit

> 2
Enter the ahnentafel number in base-10: 15
ahnentafel number in binary: 1111
family relation: mother's mother's mother
generations back: 3

1) description
2) ahnentafel number (base 10)
3) ahnentafel number (base 2)
4) relation (e.g. mother's mother's father)
5) exit

> 3
Enter the ahnentafel number in binary: 1100100
base-10 value: 100
family relation: mother's father's father's mother's father's father
generations back: 6

1) description
2) ahnentafel number (base 10)
3) ahnentafel number (base 2)
4) relation (e.g. mother's mother's father)
5) exit
```

```
> 4
Enter family relation (e.g.) "father's mother": mother's father's mother
ahnentafel number in binary: 1101
ahnentafel number in base-10: 13
generations back: 3

1) description
2) ahnentafel number (base 10)
3) ahnentafel number (base 2)
4) relation (e.g. mother's mother's father)
5) exit

> 5
Goodbye.
```

As an example of output for command line parameters, look at the file `output2.txt`.

If you enter the ahnentafel number of "1", or the relation of "self", the output should be as follows:

```
$ ahnentafel

1) description
2) ahnentafel number (base 10)
3) ahnentafel number (base 2)
4) relation (e.g. mother's mother's father)
5) exit

> 2
Enter the ahnentafel number in base-10: 1
ahnentafel number in binary: 1
self
generations back: 0

1) description
2) ahnentafel number (base 10)
3) ahnentafel number (base 2)
4) relation (e.g. mother's mother's father)
5) exit

> 3
Enter the ahnentafel number in binary: 1
base-10 value: 1
self
generations back: 0
```

```
1) description
2) ahnentafel number (base 10)
3) ahnentafel number (base 2)
4) relation (e.g. mother's mother's father)
5) exit

> 4
Enter family relation (e.g.) "father's mother": self
ahnentafel number in binary: 1
ahnentafel number in base-10: 1
```

3.3 Command Line Parameters

Your program should also accept command line parameters.

When accepting command line parameters, your program will run only once and will not display the menu.

If a base 10 value is entered, your program should output as in Item '2'.

If a binary value is entered, your program should output as in Item '3'.

If a relation is entered (e.g. father's mother's father), your program should output as in Item '4' above.

A binary number, when entered at the command line will always end with a "b", this way you can differentiate between some decimal numbers that only use 1s and 0s and an actual binary number.

In each of these cases, after printing the output, the program will terminate, returning `EXIT_SUCCESS`.

When entering command line parameters, if the data you wish to enter involves an apostrophe, you will need to escape the apostrophe since it is entered using the single quote character. This is because the shell interprets the single quote character as the start of a string.

3.4 Error Checking

For the menu part of your program, you should ensure that a valid selection was made from the list (1-5). If something other than that was entered, your program should print "Unknown operation!" to the `stderr` stream and go back to the menu.

When entering a binary number, your program should check that the number contains only 0s and 1s. If this is not the case, your program should print "Invalid string!" to `stderr` and go back to the menu.

When entering data at the command line, you should also check if the number is a valid binary number. If the user enters a decimal number such as "123ab5", it is ok to treat this as the number 123, which is what `strtol()` does.

3.5 Notes

It is recommended to use `fgets()` to read in input for the menu-based component of this program.

Of course you will not need to use `fgets()` when reading command line arguments.

3.5.1 Covert from base 10 to binary

There are a number of ways to convert a number from base 10 to base 2. Here is one algorithm that works:

1. divide input by 2 and store remainder
2. store quotient back to the input number
3. repeat until quotient becomes 0
4. binary number is the result of remainders in reverse order

3.5.2 Convert binary to base 10

You can use a similar approach when going in the other direction, or there is actually a function that you've already used, that will do the conversion for you automatically.

3.5.3 How to calculate the number of generations back

One of the outputs of your program should be a message displaying how many generations back a given individual is from the starting individual. There are a few ways to do this:

- Calculate the log base 2 of the ahnentafel number (in decimal) and then take the floor of this value. Look in the math.h library for both of these functions.

  OR

- Take the number of binary digits minus one. This simple approach works because each binary digit represents a new generation.

3.5.4 `strtok()`

One way to parse input for Item '4', that may have spaces, is to use the string tokenizer function `strtok()` which takes two parameters: a string (array of chars) that we wish to split up, and a delimiter that we use to separate the string into tokens. `strtok()` is a little unusual, in that you'll need to have several calls to the function to break up the string in the desired way. For this reason, the first call usually takes place outside a loop and subsequent calls inside a loop.

In order to use `strtok()` you need to `#include` the `string.h` library.

Here is a template, you can use in your program:

```
char str[] = "A string, with spaces.";          // string - array of chars
char *pch;                                      // pointer to keep track of str
printf( "Split str \"%s\" into tokens:\n",str );
pch = strtok( str, " ,." );                     // first call

while ( pch != NULL ) {                         // continue until end of str
    printf( "%s\n",pch );
    pch = strtok( NULL, " ,." );                // subsequent call
}
```

3.5.5 Data Types and Sizes

You should use appropriate data types and large enough fixed-sized arrays to accomplish the objectives of this project. Your ahnentafel number can be up to 32 binary digits, or the range of an unsigned int.

When you read in the relation (e.g. mother's father) you should store this string in an array large enough to store the corresponding largest ahnentafel number. Remember that the space characters also needs to be accounted for.

4 Grading

This assignment is worth a total of 100 points, distributed as follows:

- 10% Designing the code so that it is structured in a modular fashion with good naming and types (e.g. use of `#define`), and reasonable functional decomposition.
- 25% Implementing base 10 conversion (Menu Item '2')
- 25% Implementing base 2 conversion (Menu Item '3')
- 25% Implementing relation (Menu Item '4')
- 15% Command line options and remaining menu implemented properly

The following penalties will be enforced:

- -50% of points earned for errors/warnings during compilation/execution.
- -20% of points earned for memory leaks.
- -20% of points earned for no/incomplete documentation.

5 Submission

Submit `project1.zip` to the myCourses dropbox for this project. This file should include:

- `ahnentafel.c`: implementation of the program according the the above specification.
- any other source/header files you created for this assignment.