ECS 30 **Homework #9 (50 points)** **Winter 2013**

Due: Monday, March 18th. Written: 4pm in 2131 Kemper. Programs: 11:55pm using handin to cs30, p9 directory. Filenames: main.c, main.h, file.c, file.h, search.c, search.h, vector.c, vector.h, Makefile, maze.c, and binary.c.

Written (4 points)

Written (4 points): pp. 562-563: 4, 7. 2 points each.

p. 562 #4
Write a recursive C function that accumulates the sum of the values in an *n*-element array.

p. 563 #7
Write a recursive function that returns the position of the last nonblank character of a string. You may assume that you are working with a disposable copy of the string.

Zyante: 1.1-1.3, 4.8, 6.15, 10.1-10.7.

Programming (46 points, 45 minutes)

All programs should be able to compile with no warnings when compiled with the –Wall option. Your Makefile must use the –Wall option on all gcc lines. We will be taking points off for warnings. You should put your name(s) in a comment on the first line of each file. You will find my executables as well testing files in ~ssdavis/30/p9. User inputs are in **bold**. The prompts and output format of each program must match the examples exactly.

1.) pp. 564 #5 (12 points, 12 minutes) Filename: maze.c

"Write a function that accepts an 8 by 8 array of characters that represents a maze. Each position can contain either an X or a blank. Starting at position (0,1), list any path through the maze to get to location (7, 7). Only horizontal and vertical moves are allowed. If no path exists, write a message indicating there is no path.

Moves can be made only to locations that contain a blank. If an X is encountered, that path is blocked and another must be chosen. Use recursion."

Addition specifications: The two dimensional array will be stored in a file. The filename will be passed as a command line parameter to main(). Instead of spaces, the letter O will indicate an open position. The only loop allowed in the program is the one to read in the maze from the file. The recursive function, find_path(), may only call itself and printf(). Hints: To avoid an infinite recursion, when you first come to a location in the maze, you should mark it with an 'X'. Just because you must print them out starting at (0,1), does not mean that your recursion must start there.

```
[ssdavis@lect1 p9]$ cat maze1.txt
XOXXXXXX
XOXXXXXX
XOOOOXXX
XXXXOXXX
XXXXOOXX
XXXXXOXX
XXXXXOOO
XXXXXXXO
[ssdavis@lect1 p9]$ maze.out maze1.txt
(0, 1)
(1, 1)
(2, 1)
(2, 2)
(2, 3)
(2, 4)
(3, 4)
(4, 4)
(4, 5)
(5, 5)
(6, 5)
(6, 6)
(6, 7)
(7, 7)
[ssdavis@lect1 p9]$ cat maze2.txt
XOXOOOOO
OXOOOOOO
OOOOOOOO
OOOOOOOO
OOOOOOOO
OOOOOOOO
OOOOOOOO
OOOOOOOO
[ssdavis@lect1 p9]$ maze.out maze2.txt
No path was found.
```

```
[ssdavis@lect1 p9]$ cat maze3.txt
XOOOOOOO
OXXXXXXO
OOOOOOOO
XOXOXXXX
XOXOXOOO
XOXOXOXO
XOXOXOXO
XOXOOOXO
[ssdavis@lect1 p9]$ maze.out maze3.txt
(0, 1)
(0, 2)
(0, 3)
(0, 4)
(0, 5)
(0, 6)
(0, 7)
(1, 7)
(2, 7)
(2, 6)
(2, 5)
(2, 4)
(2, 3)
(3, 3)
(4, 3)
(5, 3)
(6, 3)
(7, 3)
(7, 4)
(7, 5)
(6, 5)
(5, 5)
(4, 5)
(4, 6)
(4, 7)
(5, 7)
(6, 7)
(7, 7)
```

2.) p. 565 #7 (10 points, 10 minutes) Filename: binary.c

"Write a recursive function that displays all the binary (base 2) numbers represented by a string of xs, 0s, and 1s. The xs represent digits that can be either 0 or 1. For example, the string 1x0x represents the numbers 1000, 1001, 1100, 1101. The string xx1 represents 001, 011, 101, 111. *Hint:* Write a helper function **replace_first_x** that builds two strings based on its argument. In one, the first x is replaced by a 0, and in the other by a 1. The set function **is_element** may be useful too."

Additional comments: My program has only two functions, main() and display(), but your display() may call other functions. The number will be no longer than 79 digits. *Hint:* Start with short strings, i.e., "0", "1", "x", "01", "00", "0x","1x", "xx", and reason out what would be necessary for them.

```
[ssdavis@lect1 p9]$ binary.out
Binary number: 1xx01
10001
10101
11001
11101
[ssdavis@lect1 p9]$
```

```
[ssdavis@lect1 p9]$ binary.out
Binary number: 11x0x11xx
110001100
110001101
110001110
110001111
110011100
110011101
110011110
110011111
111001100
111001101
111001110
111001111
111011100
111011101
111011110
111011111
[ssdavis@lect1 p9]$
```

3) (24 points, 30 minutes) Add the ability to print the family tree of a person to the Family program. This will entail adding a menu, reworking find_children(), and adding a recursive function, named printTree to print the family tree. You may assume that user will use the menu properly. You must an enum for the three possible values in the menu, 0, 1, and 2. Since the enum will only be used in main.c, place its definition at the top of main.c.

Suggested order of development:

As usual, your code should compile without warnings and run perfectly after each step.

1. Write a run() function that has a loop that prints a menu, and calls find_children() as it is currently written. Call run() from main(), and eliminate the call to find_children() from main(). Write the enum at the top of main.c, and use its identifiers in run(). Add a stub for printTree() that takes the same parameters and find_children().

2. In find_children(), eliminate the loop based on "Done."
3. Change find_children() and print_children() so the children are printed on the same line as the named individual with proper comma placement, and a colon after the named individual when there are children.
4. Move asking for name and finding the name_index from find_children() to run(). Change find_children() and printTree() to take an additional char* parameter that is the ID of the named individual.
5. Call find_children() from printTree().
6. Append to printTree() the first lines in find_children() that get to the point of assigning family.FAM. This should compile, but there will be warnings about unused variables.
7. In printTree(), if family.FAM is not NULL, then search for the family. If the family is found (and it should be), then create a for-loop that will call printTree() with the ID of each of its child. You are done!

```
[ssdavis@lect1 p9]$ family.out smith.ged

Menu
0. Done.
1. Find children.
2. Print family tree.
Your choice: 1
Please enter a name: Edwin
Edwin Michael Smith: Mason Michael Smith, Amber Marie Smith

Menu
0. Done.
1. Find children.
2. Print family tree.
Your choice: 2
Please enter a name: Martin Smith
Martin Smith: Hanna Smith, Ingar Smith, Ingeman Smith, Martin Smith
Hanna Smith never married.
Ingar Smith never married.
Ingeman Smith never married.
Martin Smith: Magnes Smith, Emil Smith, Gustaf Smith , Sr.
Magnes Smith had no children.
Emil Smith never married.
Gustaf Smith , Sr.: Kirsti Marie Smith, Astrid Shermanna Augusta Smith, Hjalmar Smith, Hjalmar
Smith, Gus Smith, Carl Emil Smith, Hans Peter Smith
Kirsti Marie Smith had no children.
Astrid Shermanna Augusta Smith had no children.
Hjalmar Smith never married.
Hjalmar Smith: John Hjalmar Smith, Marjorie Lee Smith
John Hjalmar Smith: Marjorie Alice Smith, Edwin Michael Smith
Marjorie Alice Smith never married.
Edwin Michael Smith: Mason Michael Smith, Amber Marie Smith
Mason Michael Smith never married.
Amber Marie Smith never married.
Marjorie Lee Smith never married.
Gus Smith had no children.
Carl Emil Smith never married.
Hans Peter Smith had no children.

Menu
0. Done.
1. Find children.
2. Print family tree.
Your choice: 2
Please enter a name: Edwin
Edwin Michael Smith: Mason Michael Smith, Amber Marie Smith
Mason Michael Smith never married.
Amber Marie Smith never married.

Menu
0. Done.
1. Find children.
2. Print family tree.
Your choice: 0
[ssdavis@lect1 p9]$
```