ECS 30 **Homework #8 (96 points)** **Fall 2013**

Due: Friday, December 6th. Written: 4 pm in 2131 Kemper. Program: 11:59pm using handin to cs30, p8 directory. Only if you handin the program by Wednesday, December 4th at 11:59pm will you receive a grade sheet for the program on Friday, December 6th. Otherwise you will receive your grade sheet on Tuesday, December 10th at the final, and you will have no opportunity to edit your program in my office.
Filenames: maze.c, binary.c, rsvp.c, vector.c, vector.h, course.c, course.h, student.c, student.h, and Makefile.

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.

Programming (92 points, 92 minutes)

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 p8]$ cat maze1.txt
XOXXXXXX
XOXXXXXX
XOOOOXXX
XXXXOXXX
XXXXOOXX
XXXXXOXX
XXXXXOOO
XXXXXXXO
[ssdavis@lect1 p8]$ 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 p8]$ cat maze2.txt
XOXOOOOO
OXOOOOOO
OOOOOOOO
OOOOOOOO
OOOOOOOO
OOOOOOOO
OOOOOOOO
OOOOOOOO
[ssdavis@lect1 p8]$ maze.out maze2.txt
No path was found.
```

```
[ssdavis@lect1 p8]$ cat maze3.txt
XOOOOOOO
OXXXXXXO
OOOOOOOO
XOXOXXXX
XOXOXOOO
XOXOXOXO
XOXOXOXO
XOXOOOXO
[ssdavis@lect1 p8]$ 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, 110, 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 p8]$ binary.out
Binary number: 1xx01
10001
10101
11001
11101
[ssdavis@lect1 p8]$
```

```
[ssdavis@lect1 p8]$ binary.out
Binary number: 11x0x11xx
110001100
110001101
110001110
110001111
110011100
110011101
110011110
110011111
111001100
111001101
111001110
111001111
111011100
111011101
111011110
111011111
[ssdavis@lect1 p8]$
```

3.) RSVP continued (70 points, 70 minutes)

The program should be able to compile with no warnings when compiled with the –Wall option. You should put your name(s) in a comment on the first line of each file. The prompts, and output format and values of each program must match the examples exactly, except that the values of calculated floating point values may be slightly different. All floating point values should be stored in doubles, not floats. Your main() must be the first function defined in rsvp.c, so you will have to provide prototypes for your other functions in rsvp.c at the top of that file. You may assume that the user will make entries correctly, unless noted otherwise. You will find my executable in ~ssdavis/30/p8 in the CSIF. Note that these are not accessible from the web.

For this assignment, you extend your WebSis program to handle student requests to add and remove course from their schedules. Your program will no longer have the parallel arrays for the course information. Instead, your implementation will now use a dynamically allocated array of typedeffed structs. Your program will also have another dynamically allocated array of typedeffed structs to hold the information about student schedules. Further specifications:

1. Course struct
   1.1. Each Course struct will hold the CRN, subject, and course for an individual offering.
   1.2. Since the length of all the strings involved are fairly consistent, you may use appropriate constants for your char arrays instead of dynamically allocating.
   1.3. The array of these structs should be initially sized to 100, and then double in size when needed.
2. Student struct
   2.1. Each Student struct will hold first name, last name, SID, an array of CRNs, and a count of classes.
   2.2. The first name, and last name will be pointers to dynamically allocated arrays. The length of the arrays must be based on their contents, and not some constant.
   2.3. A student will take no more than five course.
   2.4. The array of these structs must be dynamically allocated 10, and then double in size when needed.
3. Adding a course.
   3.1. If the SID cannot be found in the students array, then the program should notify the student, and return to the main menu.
   3.2. If a student attempts to add a course when they already have five course, the program should notify the student, and refuse to continue with the addition process.
   3.3. If a student attempts to add a course they already have, then the program should notify the student, and NOT add the course a second time.
   3.4. Do not save the changes to the students file.
4. Removing a course
   4.1. If the SID cannot be found in the students array, then the program should notify the student, and return to the main menu.
   4.2. If the student attempts to remove a course that is not in their schedule then the program should notify the student.
   4.3. Do not save the changes to the students file.
5. Course by CRN
   5.1. Besides the information about the course, the program should list all of the students that are taking the course.
6. Course by subject
   6.1. This will list all of the information for all of the course with that subject. It does not list the information about the students.
7. Most constants should be #defined, except 0, and menu items.
8. main() must still have only variable declarations, function calls, and a return statement.
   8.1. The first command line parameter will be the name of the course file, and the second parameter will be the name of the student file.
9. Makefile must use –g and –Wall on all gcc lines. You must have pairs of lines for each source code file.
10. Hints and suggestions:
    10.1. As before, you should do top down design with stubs. Implementing, and testing each function one at a time. I suggest you leave your original code from P7, and replace function calls with new function calls one by one. When re-working a P7 function, you may wish to comment out either individual lines, or whole blocks of code so that it will compile despite have different parameters.
    10.2. To avoid circular references in your header files, you should have all the functions that mention both Student and Course in course.c, other than display_info().
    10.3. For students.csv, watch out for the '\n' in strtok for Ms Saeteurn.

```
[ssdavis@lect1 p8]$ cat students.csv
Boyd,An,903-04-2620,53583,54678,59414
Chen,Andrew,995-52-6987,57056,56361,60360,58819
Delgado,Chetan,992-63-3996,57056,60359,58820
Hong,Frederick,992-84-9667,61802,57058,60360,58841
Lee,Daniel,995-59-4469,61750,60189,60360,58890
Li,Garret,995-65-9809,60853,60295,60360,59363
Mangan,Janet,995-59-4760,60855,61367,59383
McCullough,Jason,995-83-8060,51909,61626,61637,60364
Ng,Joey,995-46-4586,57056,61670,61674,60364
Park,Komal,995-50-9519,61702,59420
```

```
Qi,Kyle,995-66-0733,52375,61703,61735,60193
Robinson,Lucia,995-45-2022,61743,60199
Saeteurn,Nancy,995-59-6454,60360
San Emeterio Nateras,Nicholas,995-16-3033,53415,61745,61807,60364
Shah,Oscar,995-45-2828,53658,61784
Tafolla,Peter,995-78-6086,53694,61703,60360,60857
Tat,Ross,995-53-2052,57056,61703,60851,60860
Tiru,Ryan,993-96-8709,53878,61703,61368,61284
Waters,Tairi,992-00-0000,53889
Zuniga,Ying Bei,994-19-7964,53890,61877,57069,61290
[ssdavis@lect1 p8]$
[ssdavis@lect1 p8]$ rsvp.out summersession_I.html students.csv

RSVP Menu
0. Done.
1. Find by CRN.
2. Find by subject.
3. Add course.
4. Remove course.
Your choice (0 - 4): 1
Please enter a CRN: 40000
CRN 40000 not found.

RSVP Menu
0. Done.
1. Find by CRN.
2. Find by subject.
3. Add course.
4. Remove course.
Your choice (0 - 4): 5
Your choice is outside the acceptable range.  Please try again.

RSVP Menu
0. Done.
1. Find by CRN.
2. Find by subject.
3. Add course.
4. Remove course.
Your choice (0 - 4): 1
Please enter a CRN: 57056
MAT 016B
Chen Andrew
Delgado Chetan
Ng Joey
Tat Ross

RSVP Menu
0. Done.
1. Find by CRN.
2. Find by subject.
3. Add course.
4. Remove course.
Your choice (0 - 4): 2
Please enter a subject: AAA
No course were found for AAA.
```

```
RSVP Menu
0. Done.
1. Find by CRN.
2. Find by subject.
3. Add course.
4. Remove course.
Your choice (0 - 4): 2
Please enter a subject: EEC
61695 EEC 180A
61696 EEC 180A

RSVP Menu
0. Done.
1. Find by CRN.
2. Find by subject.
3. Add course.
4. Remove course.
Your choice (0 - 4): 3
Add Course
Please enter the SID of the student: 995-16-3033
Please enter the CRN: 40000
There is no course with CRN #40000

RSVP Menu
0. Done.
1. Find by CRN.
2. Find by subject.
3. Add course.
4. Remove course.
Your choice (0 - 4): 3
Add Course
Please enter the SID of the student: 995-16-3033
Please enter the CRN: 54625

RSVP Menu
0. Done.
1. Find by CRN.
2. Find by subject.
3. Add course.
4. Remove course.
Your choice (0 - 4): 3
Add Course
Please enter the SID of the student: 995-16-3033
You are already taking 5 courses.

RSVP Menu
0. Done.
1. Find by CRN.
2. Find by subject.
3. Add course.
4. Remove course.
Your choice (0 - 4): 4
Remove Course
Please enter the SID of the student: 995-16-3033
Current course: 53415 61745 61807 60364 54625
Please enter the CRN: 61807
```

```
RSVP Menu
0. Done.
1. Find by CRN.
2. Find by subject.
3. Add course.
4. Remove course.
Your choice (0 - 4): 4
Remove Course
Please enter the SID of the student: 995-16-3033
Current course: 53415 61745 54625 60364
Please enter the CRN: 53415

RSVP Menu
0. Done.
1. Find by CRN.
2. Find by subject.
3. Add course.
4. Remove course.
Your choice (0 - 4): 0
[ssdavis@lect1 p8]$
```