ECS 30 **Homework #7 (50 points)** **Fall 2013**

Due: Monday, November 25th . Program: 11:59pm using handin to cs30, p7 directory.
Filenames: rsvp.c. vector.c, vector.h, course.c, course.h, and Makefile. Executable name: rsvp.out
Written: (10 points) pp. 614-615: 1, 3; pp. 654-655: 4, 6; p. 695: 4, 5, 6, 10, 11, 12.

p. 614 #1
Define a structure type called `subscriber_t` that contains the components `name, street_address, and monthly_bill` (i.e., how much the subscriber owes).

p. 615 #3
From p. 613 #2, there was a typedef struct `olympic_t`, and a declaration `olympic_t competiton`. How would you call a function `scan_olympic()` passing `competition` as an output argument?

p. 654 #4
What are the characteristics of a binary file?

pp. 655 #6
What is a file pointer?

p. 695 #4
How does the C compiler know whether to look for an included file in the system directory or in the program's directory?

p. 695 #5
Compare the execution of the macro call: `MAC(a,b);`

to the execution of an analogous function call: `mac(a,b);`

Which of the following two calls is sure to be valid and why?

`mac(++a, b);` or `MAC(++a, b);`

p. 695 #6
When you write the body of a macro definition, where should you use parentheses?

p.695 #10
Why is the argument value 1 used much more often than the argument value 0 in calls to the `exit()` function?

p. 695 #11
Describe the purpose of the "**`defined`**" operator.

p. 695 #12
When function `main()` of a C program has a non-void parameter list, why is the valu of its first parameter never less than 1?

Program (40 points, 40 minutes)

This program should be able to compile with no warnings when compiled with the –Wall option, e.g. `gcc –Wall rsvp.c`. 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. Your main() must be the first function defined in the rsvp.c file, so you will have to provide prototypes for your other functions at the top of the file. You may assume that the user will make entries of the correct type. You will find my executable in ~ssdavis/30/p7 in the CSIF. Note that these are not accessible from the web.

For this assignment, and the last assignment, you will be writing a program that provides some of the functionality of SISWeb. This program will provide the basis for the program in the final assignment. This program should be able to read and parse either summersession_I.html. or summersession_II.html. You may copy these files from ~ssdavis/30/p7.

Here are the specifications:

1. We will limit ourselves to courses that are not discussions/labs, and have five-digit CRNs.
2. Header files contain the proper preprocessor directives for a large project, i.e., #ifndef …, and the prototypes for the corresponding source code files.
3. You will store the CRNs, subjects, and course numbers/letters in three dynamically allocated parallel arrays.
   3.1. Since the arrays will be sized larger than needed, you will need to keep track of their size, and the number elements current stored in them. size and count are good names for such variables.
4. rsvp.c
   4.1. main() will contain only variable declarations, function calls, and a return statement. The name of the course file will be passed as the program's command line parameter.
   4.2. get_choice() displays the menu, and queries the user for a choice until the user provides a valid choice.
   4.3. display_info() calls get_choice, find_CRN(), and find_subject() repeatedly.
   4.4. You should have different functions for different aspects of interacting with the user, and the arrays.
5. course.c
   5.1. Parsing involves detecting patterns, and then writing code that can differentiate desired data from undesired.
      5.1.1. Patterns can be related to location, constant content, and/or tokenized position. The patterns for this part of the program are fairly easy to determine.
      5.1.2. In the context of this program, you will need to detect patterns to differentiate between useful lines and useless lines in the file.
      5.1.3. Once you find a useful line, you will have further parse it to retrieve the CRN, subject, and course from it.
   5.2. read_courses() will be responsible for opening, reading, and parsing the file passed as the command line parameter to the program.
      5.2.1. If the file cannot be opened, the program should report it, and then exit(1). exit() is in stdlib.h
      5.2.2. It will call initialize(), and repeatedly call resize().
      5.2.3. The sections of the file you will be reading are tab delimited so strtok() will be useful, though the '^'s are troublesome.
   5.3. find_CRN() will query the user for a CRN, and then display the course if it is found.
   5.4. find_subject() will query the user for a subject, and then display the courses of that subject if found.
6. vector.c
   6.1. This file will have three functions.
   6.2. initialize() will dynamically allocate each of the three arrays to size 100.
      6.2.1. CRNs may be stored as ints.
      6.2.2. Each element of the subjects array will point to a dynamically allocated char[4] array.
      6.2.3. Each element of the courses array will point to a dynamically allocated char[6] array.
      6.2.4. Since you will be changing to where the pointers point, this will involve triple pointers for subjects and courses!
   6.3. resize() will create three arrays, each twice the size of the corresponding current size, and then copies the old array's elements to the beginning of the new arrays. After copying the old arrays, aspects of them should be freed.
      6.3.1. This will involve triple pointers too!
   6.4. deallocate() will free all dynamically allocated arrays.
      6.4.1. There is no need to pass the address of the double pointers, nor the address of size, for this function.
7. Makefile
   7.1. It must create an object file for each source code file.
   7.2. You must use the –Wall and –g options on all lines invoking gcc.
   7.3. It must have a clean: option that uses rm –f to explicitly remove the files created by the Makefile, i.e., vector.o, rsvp.o, course.o, and rsvp.out.
8. Suggestions and hints
   8.1. You should make use of top down design, and write stubs.
   8.2. Write only one function at a time then compile and run it until it is error free. Now that you know gdb, you can more easily check variable values at the end of a function.
   8.3. Remember that the [] operator has higher precedence than the * operator so when working with triple pointers you will need to use parentheses to ensure that the dereferencing occurs before the indexing, e.g., (*subjects)[i].
   8.4. Remember to compare your program against mine using diff. To do that, create a text file (say, input.txt) corresponding to your input and call both programs, then compare:

```
rsvp.out summersession_I.html < input.txt > my_output.txt
~ssdavis/30/p7/rsvp.out summersession_I.html < input.txt > seans.txt
  diff -bB my_output.txt seans.txt

You can also use the nice graphical utility meld:
  meld my_output.txt seans.txt
```

Here is part of summersession_I.html:

```
African American & African Std (AAS)

61753    AAS   050   COM   A01   Black Images   4.0   Harrison, MF   TR 1210-0150P  WELLMN   226   25
                           TR 0210-0350P  OLSON 163
61754    AAS   050   COM   A02   Black Images   4.0   Harrison, MF   TR 1210-0150P  WELLMN   226   25
                           TR 0410-0550P  OLSON 163
51010    AAS   100   COM   A01   Ethnicity In US   4.0   Harrison, MF   TWR   1000-1140A  CHEM  176   25
                           W  0410-0550P  BAINER   1130
51011    AAS   100   COM   A02   Ethnicity In US   4.0   Harrison, MF   TWR   1000-1140A  CHEM  176   25
                           F  1210-0150P  BAINER   1130
61775    AAS   181   LED   001   Hip Hop in Urban America   4.0   The Staff   MTWR  0610-0750P  OLSON 118   50

Agricultural & Resource Econ (ARE)

51962 ^  ARE   018   LEC   001   Business Law   4.0   The Staff   MTWR  1000-1140A  OLSON 146   60
52047 ^  ARE   100A  COM   A01   Intermed Microeconomics 4.0   The Staff   MTW   1210-0150P  HARING   1227  100
                           R  1210-0150P  HARING   1227
```

```
[ssdavis@lect1 p7]$ rsvp.out summersession_I.html

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

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

RSVP Menu
0. Done.
1. Find by CRN.
2. Find by subject.
Your choice (0 - 2): 1
Please enter a CRN: 61753
AAS 050

RSVP Menu
0. Done.
1. Find by CRN.
2. Find by subject.
Your choice (0 - 2): 1
Please enter a CRN: 61000
CRN 61000 not found.

RSVP Menu
0. Done.
1. Find by CRN.
2. Find by subject.
Your choice (0 - 2): 2
Please enter a subject: AAS
61753 AAS 050
61754 AAS 050
51010 AAS 100
51011 AAS 100
61775 AAS 181

RSVP Menu
0. Done.
1. Find by CRN.
2. Find by subject.
Your choice (0 - 2): 2
Please enter a subject: ARE
51962 ARE 018
52047 ARE 100A
61657 ARE 100A
61854 ARE 115A
52048 ARE 115B
52049 ARE 115B
52050 ARE 142
52052 ARE 155
52055 ARE 171A
61659 ARE 171A
61815 ARE 143

RSVP Menu
0. Done.
1. Find by CRN.
2. Find by subject.
Your choice (0 - 2): 2
Please enter a subject: ABC
No courses were found for ABC.

RSVP Menu
0. Done.
1. Find by CRN.
2. Find by subject.
Your choice (0 - 2): 0
[ssdavis@lect1 p4]$
```