ECS 30 **Homework #6 (60 points)** **Winter 2013**

Due: Wednesday, February 20th. Written: 4pm in 2131 Kemper. Programs: 11:55pm using handin to cs30, p6 directory. Filenames: main.c, prime.c, typescript, power.c, resistor.c, count.c

gdb Tutorial (10 points)

The interactive gdb tutorial is available online from the course website. Extra TAs will be in 67 and 75 Kemper 6-8pm on Thursday, and Tuesday to assist anyone with the tutorial. You should submit main.c, prime.c, and your typescript from this tutorial. Each person must do the tutorial individually.

Written (8 points)

Written (8 points) pp. 511-512 1, 2, 3, 4, 5, 8, 9, 10.

"Refer to these declarations when determining when determining the effect of the statements in Questions 1-4.

```
char s5[5], s10[10], s20[20];
char aday[7] = "Sunday";
char another[9] = "Saturday";
```

```
1. strncpy(s5, another, 4);  s5[4] = '\0';
```

```
2. strcpy(s10, &aday[3]);
```

```
3. strlen(another);
```

```
4. strcpy(s20, aday); strcat(s20, another);
```

5. Write a function that pads a variable-length string with blanks to its maximum size. For example, if `s10` is a ten-character array currently holding the string "screen", **`blank_pad`** would add three blanks (one of which would overwrite the null character) and finish the string with the null character. Be sure your function would work if no blank padding were necessary.

8. Which one of the following would call somefun only if the string values of character arrays a and b were equal?

```
a. if (strcmp(a, b))
     somefun();

b. if(strcmp(a, b) == 0)
     somefun();
```

```
c. if(a == b)
     somefun();

d. if(a[] == b[])
     somefun();
```

9. What does this program fragment display?

```
char x[80] = "gorilla";
char y[80] = "giraffe";
strcpy(x, y);
printf("%s %s\n", x, y);
```

```
a. gorilla giraffe
b. giraffegorilla gorilla
```

```
c. gorilla gorilla
d. giraffe giraffe
```

10. What does this program fragment display?

```
char x[80] = "gorilla";
char y[80] = "giraffe";
strcat(x, y);
printf("%s %s\n", x, y);
```

```
a. gorillagiraffe giraffe                    c. gorilla gorilla
b. giraffegorilla gorilla                    d. giraffe giraffe
```

Zyante: 5.7, 5.8, 5.10, 8.3, 8.4

Programming (42 points)

All programs should be able to compile with no warnings when compiled with the –Wall option, e.g. gcc –Wall taxes.c. Beginning this week we will be taking points off for warnings. You should put your name(s) in a comment on the first line of each file. When you are to write your own functions, since main() should be the first function in each file, you will need to provide a prototype above main() for each function you write. You will find my executables as well testing files in ~ssdavis/30/p6. The prompts and output format of each program must match the examples exactly. User inputs are in **bold**.

#1. Filename: power.c Adapted from pp. 450-451 #15 of the text (8 points, 9 minutes)

Peabody Public Utilities traces the status of its power service throughout the city with a 3 x 4 grid in which each cell represents power service in one sector. When power is available everywhere, all grid values are 1. A grid value of 0 indicates an outage somewhere in the sector.

Write a program that inputs a grid from a file and displays the grid. If all grid values are 1, then display the message "Power is on throughout the grid." Otherwise, list the sectors that have outages. Include in your program functions get_grid(), display_grid(), power_ok(), and where_off(). Function power_ok() returns true (1) if power is on in all sectors, false (0) otherwise. Function where_off() shold display the message regarding sectors experiencing outages. The 3 x 4 grid must be dynamically allocated in main().

```
[davis@lect1 p6]$ power.out
Filename: power1.txt
0 1 1 1
1 1 0 1
1 1 1 1

Power is off in sectors:
      (0,0)
      (1,2)
[davis@lect1 p6]$
[davis@lect1 p6]$ power.out
Filename: power2.txt
1 1 1 1
1 1 1 1
1 1 1 1

Power is on throughout the grid.
[davis@lect1 p6]$
```

5.) Filename: resistor.c Adapted from pp. 512-513 #2 in the text (13 points, 15 minutes) We will compile with –lm.

A resistor is a circuit device designed to have a specific resistance value between its ends. Resistance values are are expressed in ohms (Ω) or kilo-ohms (kΩ). Resistors are frequently marked with three colored bands that encode their resistance values. The first two bands are digits, and the third is a power of ten multiplier.

The table below shows the meanings of each band color. For example, if the first band is green, the second is black, and the third is orange, the resistor has a value of 50 x 10^3Ω or 50kΩ. The names of the colors can be stored in an array of 10 strings. If they are placed in the correct order, i.e. "black", then "brown" etc., then their index value will also be their digit value.

Write a program that prompts for the colors of the three bands, and then displays the resistance in kilo-ohms. Include a function named search() that takes three parameters—the list of strings, the size of the list, and a target string, and returns the subscript of the list element that matches the target or returns -1 if the target is not in the list."

Color	Value as Digit	Value as Multiplier
black	0	1
brown	1	10
red	2	10^2
orange	3	10^3
yellow	4	10^4
green	5	10^5
blue	6	10^6
violet	7	10^7
gray	8	10^8
white	9	10^9

```
[davis@lect1 p6]$ resistor.out
Enter the colors of the resistor's three bands, beginning with
the band nearest the end.  Type the colors in lower case letters
only, NO CAPS.
Band 1 => green
Band 2 => black
Band 3 => yellow
Resistance value:  500 kilo-ohms
Do you want to decode another resistor?
=> y
Enter the colors of the resistor's three bands, beginning with
the band nearest the end.  Type the colors in lower case letters
only, NO CAPS.
Band 1 => brown
Band 2 => vilet
Band 3 => gray
Invalid color: vilet
Do you want to decode another resistor?
=> n
[davis@lect1 p6]$
```

6.) Filename: count.c (21 points, 25 minutes)

Write a program that reads a file, displaying a count of the total number of words in the file, the number of words with one letter, two letters, and so on, the number of unique words (ignoring the case of the word), and the final size of its "unique" array. The maximum length of a word is 30 characters, not including the '\0'. Words are composed only of consecutive alphabetic characters. Use tolower(), and isalpha() of ctype.h for constructing your words in lower case letters.

In order to keep track of the unique words seen so far you must use a dynamically allocated array of strings, named unique. Each char array that unique points to must be dynamically allocated based on the size of the word it will contain. The initial size of unique will be 10 char*'s. When the program encounters a word that is not in unique, it will allocate a char array to hold the new word, append the pointer to the new word to the list in unique, and increment a variable named count that keeps track of how many items are in unique.

If the count variable becomes equal to the size of unique, then the program will call the function resize(). resize() creates a new array, named temp, with twice as many char*'s as the current unique, copies the char*'s from unique to temp (don't use strcpy(), just assign), then frees() the old unique (but not its arrays of chars), and returns the address held in temp. Though resize() will take unique and the address of unique's current size as parameters, by returning temp, resize() avoids having triple pointers. You will simply assign the unique variable in the calling function the value returned by resize().

You must call a function named free_unique() near the end of main() to deallocate ALL dynamically allocated memory.

You may assume that each string read by fscanf(…"%s"…) will contain at most one legitimate word.

```
[ssdavis@lect1 p6]$ cat gettysburg.txt
Four score and seven years ago our fathers brought forth on this continent a new nation, conceived
in liberty, and dedicated to the proposition that all men are created equal.
Now we are engaged in a great civil war, testing whether that nation, or any nation, so conceived
and so dedicated, can long endure. We are met on a great battle-field of that war. We have come to
dedicate a portion of that field, as a final resting place for those who here gave their lives that
that nation might live. It is altogether fitting and proper that we should do this.
But, in a larger sense, we can not dedicate, we can not consecrate, we can not hallow this ground.
The brave men, living and dead, who struggled here, have consecrated it, far above our poor power
to add or detract. The world will little note, nor long remember what we say here, but it can never
forget what they did here. It is for us the living, rather, to be dedicated here to the unfinished
work which they who fought here have thus far so nobly advanced. It is rather for us to be here
dedicated to the great task remaining before us - that from these honored dead we take increased
devotion to that cause for which they gave the last full measure of devotion - that we here highly
resolve that these dead shall not have died in vain - that this nation, under God, shall have a new
birth of freedom - and that government of the people, by the people, for the people, shall not
perish from the earth.
[ssdavis@lect1 p6]$ count.out
Filename: gettysburg.txt
Total words: 271  Unique words: 138  Uniques size: 160
Length Count
   1      7
   2     50
   3     60
   4     58
   5     34
   6     25
   7     15
   8      6
   9     10
  10      4
  11      2
[ssdavis@lect1 p6]$
[ssdavis@lect1 p6]$ cat rounding.c
// Author: Sean Davis
// modified to ensure fscanf(...%s...) reads only one "word" at a time

#include <stdio>

int main()
{
  double value, product, result;
  printf( "Please enter a monetary value: ");
  scanf( "%lf", &value);
  product = value / 1000000;
  result = product * 1000000;
  printf( "%16.2lf %24.10lf %34.20lf", result, result, result);
  return 0;
} // main()

[ssdavis@lect1 p6]$ count.out
Filename: rounding.c
Total words: 44  Unique words: 29  Unique's size: 40
Length Count
   1      2
   2      6
   3      2
   4      6
   5      9
   6     13
   7      4
   8      2
[ssdavis@lect1 p6]$
```