CSCE 105 Homework 5 Due November 18, 2004

Name and SSN: **KEY**

Instructor's Name: **Andrew Charles Breiner**

Question	Points	Score
1	10	
2	10	
3	10	
4	10	
5	10	
6	10	
7	10	
8	10	
9	10	
10	10	
Total:	100	

1. (10 points) If squaring 10^{-20} gives a result of zero, the type of error that has occurred is called what? (Question (2) of Self-Check on page 336 in section 7.1)

Solution:

underflow

2. (10 points) Evaluate the following expressions if **x** is 4.5, **y** is 3.1, **m** is 5, and **n** is 2.

Solution:

(a)	x / (double)m	**0.9**
(b)	x / m	**0**
(c)	(double)(n * m)	**10.0**
(d)	(double)(n / m) + y	**3.1**
(e)	(int)(x / y)	**1**

3. (10 points) Write a for loop that would print the alphabet in lowercase letters, assuming that letters have consecutive orders, use explicit conversion to go from char to int and from int to char. (Question (2) of Self-Check on page 339 in section 7.3)

Solution:

```
for(i = (int)'a'; i <= (int)'z'; i++) {
        printf("%c ", (char)i);
}
```

4. (10 points) Indicate whether each of the following type definition groups is valid or invalid. Explain the flaws in invalid definitions. (Question (2) of Self-Check on page 345 in section 7.3)

Solution:

a. typedef enum
{'A', 'B', 'C', 'D'}
letters_t;
not valid - elements are char and are not valid

b. typedef enum
{a, b, c}
letters_t;
typedef enum
{c, d}
twolet_t;
not valid - c can only be used once

c. typedef enum
{while, for, if, switch}
stmt_t;
not valid - using reserved words

5. (10 points) List and explain three computational errors that may occur in type double expressions. (Question (2) of Review Questions on page 360)

Solution:

underflow - when our fraction gets too small
overflow - when our value gets too large
cancellation - when we add two numbers and one is very large and the other is very small, then we may not get all of the small number included

6. (10 points) Write a prototype for a function find_avg that has two type double input parameters and three integer output parameters, the function has no return statement.

Solution:

void find_avg(double x, double y, int *a, int *b, int *c);

7. (10 points) Show the table of values for **x**, **y**, and **z** that is the output displayed by the following program. You will notice that the function **sum** does not follow the suggestion in the last Program Style segment of Section 6.2. You can improve this program in the programming exercise that follows. (Question (2) of Self-Check on page 295 in section 6.2)

```
#include <stdio.h>
void sum(int a, int b, int *cp);

int main(void) {
   int x, y, z;
   x = 5; y = 3;
   printf("   x   y   z\n\n");
   sum(x, y, &z);
   printf("%4d%4d%4d\n", x, y, z);
   sum(y, x, &z);
   printf("%4d%4d%4d\n", y, x, z);
   sum(z, y, &x);
   printf("%4d%4d%4d\n", z, y, x);
   sum(z, z, &x);
   printf("%4d%4d%4d\n", z, z, x);
   sum(y, y, &y);
   printf("%4d%4d%4d\n", y, y, y);
   return (0);
}
void sum(int a, int b, int *cp) {
   *cp = a + b;
}
```

Solution:

x	y	z
5	3	8
5	3	8
8	5	13
8	8	16
5	5	10

8. (10 points)
   a. Classify each formal parameter of double_trouble and trouble as input, output, or input/output.
   b. What values of x and y are displayed by this program? (Hint: Sketch the data areas of main, trouble, and double_trouble as the program executes.)

```
void double_trouble(int *p, int y);
void trouble(int *x, int *y);

int main(void) {
   int x,y;
   trouble(&x,&y);
   printf(x = %d, y = %d\n", x, y);
   return 0;
}
void double_trouble(int *p, int y) {
   int x;
   x = 14;
   *p = 2 * x - y;
}
void trouble(int *x, int *y) {
   double_trouble(x, 5);
   double_trouble(y, *x);
}
```

(Question (2) of Self-Check on page 302 in section 6.4)

Solution:

a. **double_trouble - output input , trouble - output output**

b. **x = 23, y = 5**

9. (10 points) Why would you choose to write a function that computes a single numeric or character value as a non void function that returns a result through a return statement rather than to write a void function with an output parameter? (Question (2) of Chapter Review on page 319)

Solution:

It is easier to understand and stores less memory.

10. (10 points) Present arguments against these statements: (Question (5) of Chapter Review on page 320)

Solution:

a. It is foolish to use function subprograms because a program written with functions has many more lines than the same program written without functions.

printf is a function subprogram and it saves a lot of lines of code

b. The use of function subprograms leads to more errors because of mistakes in using argument lists.

functions can be tested separately and makes the code more readable and we can now better find errors