Talen Phillips

EE 107-01

Homework #5

Due: 25MAR2013

**1)** Write a program that finds the largest in a series of numbers entered by the user. The program must prompt the user to enter numbers one by one. When the user enters 0 (zero) or a negative number, the program must display the largest nonnegative number entered. For example:

Enter a number: 60

Enter a number: 38.3

Enter a number: 4.89

Enter a number: 0

The largest number entered was 60.0

/\* Talen Phillips

\* EE107-01

\* 25MAR2013

\* "Largest Number"

\*/

#include <stdio.h>

int main (void)

{

float input,largest;

do{

printf("Enter a number: ");

scanf("%f",&input);

if (input > largest) largest = input;

// Each input is compared to the current largest

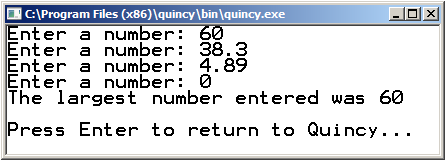
}while (input > 0); // Loop ends when input <=0

printf("The largest number entered was %g\n",largest);

// NOTE: %g was used to drop the trailing zeros.

return 0;

}



**2)** Write a program that asks the user to enter a fraction, then reduces the fraction to lowest terms. For example:

Enter a fraction: 6/12

In lowest terms: 1/2

Hint: Work with the lowest common denominator.

/\* Talen Phillips

\* EE107-01

\* 25MAR2013

\* "Fraction Simplifier"

\*/

#include <stdio.h>

int main (void)

{

int n,d,L,gcd;

/\* n is the numerator,

\* d is the denominator,

\* L is the largest of those two, and

\* gcd is the greatest common denominator

\*/

//first take the input.

printf("Enter a fraction: ");

scanf("%d/%d",&n,&d);

//then find the larger number:

if (d > n) L = d;

else L = n;

//then find the greatest common denominator:

for (gcd=L;gcd>=1;gcd--)

if ((n % gcd == 0) && (d % gcd == 0))break;

//then display the reduced fraction:

if ((d/gcd) > 1)

printf("in lowest terms: %i/%i\n",n/gcd,d/gcd);

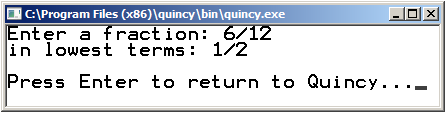
//if denominator/gcd is 1, only display numerator/gcd:

if ((d/gcd) == 1)

printf("in lowest terms: %i\n",n/gcd);

return 0;

}



**3)** The value of the mathematical constant “e” can be expressed as an infinite series, i.e.

e = 1 + 1/1! + 1/2! + 1/3! + …

Write a program that approximates “e” by computing the value of (1 + 1/1! + 1/2! + 1/3! + … + 1/n!) where n is an integer entered by the user.

/\* Talen Phillips

\* EE107-01

\* 25MAR2013

\* "e Calculator"

\*/

#include <stdio.h>

int main (void)

{

int n,x; //counter and input variables

double result = 1; //running total

double fact = 1; //factorial variable

printf("Enter a number: ");

scanf("%d",&x);

for (n=1;n<=x;n++){

fact \*= n; //increment factorial

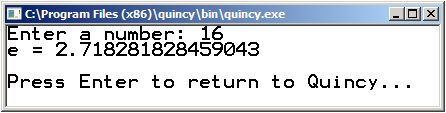
result += (1.0/fact);//add 1/fact to the total

}

printf("e = %.16g\n",result);

return 0;

}



**4)** Rewrite the following loop using only the for statement, i.e. using only -> for ( … );

for (n=0; m > 0; n++)

m /= 2;

for (n=0; m > 0; n++, m/=2);

**5)** Which one of the following statements is not equivalent to the other two (assuming the loop bodies, or statements between the curly brackets, are the same) and explain why?

(a) while(i<10) { … }

(b) for(;i<10;) { … }

(c) do { … } while(i < 10);

(c) do { … } while(i < 10); isn't equivalent to the others, because it checks the loop condition *after* the first iteration. It will always execute at least once even if (i >= 10)

**6)** What output does the following program fragment produce?

i=1;

while (i<=128)

{

printf(“%d “,i);

i \*= 2;

}

This code fragment would list all of the exponents of two from 2¹ up to 2128 in order on the same line like this:

1 2 4 8 16 32 64 128

**7)** What output does the following program fragment produce?

i = 9384;

do {

printf(“%d “,i);

i /= 10;

} while (i>0);

This code:

1) prints 9384

2) divides by 10

3) checks that the result is greater than 0

4) prints the result

5) repeats steps 2 through 4 until step 3 returns false.

Anything to the right of the decimal is omitted, and all results are printed on the same line like this:

9384 938 93 9

Note: It really doesn't matter that the initial check doesn't happen until after the first iteration, since the variable i is initiated just before the loop.

**8)** Rewrite the code in problem 6 using a for loop.

for (i=1;i<=128;i\*=2) printf("%d ",i);

**9)** Rewrite the code in problem 7 using a for loop.

for (i=9384 ; i>0 ; i/=10) printf("%d ",i);

Note: as mentioned in problem 7, the first condition check makes no difference, since the variable is initialized as part of the loop. If the value of i was dependant on something outside the loop, and the loop still needed to execute at least once, this could be accomplished with the addition of a loop counter like so:

for (i=9384 , j=0 ; (j=0)||(i>0) ; i/=10 , j++) printf("%d ",i);