Cs352 — Homework #2

Due Time: 9/16/03 (9:00PM)

Turnin ID: $cs352\_assg2$

Turnin files: Qremove.c, ExponentCheck.c, sum.c

(PS: For example, you may turnin multiple files by UNIX command "turnin $cs352\_assg1$ yourfile1 yourfile2 yourfile3". You may also turnin one file at a time by "turnin $cs352\_assg1$ $file\_you\_want\_to\_turnin$". Later turnin file will overwrite the old file which has the same filename. So if you turnin the same file multiple times, we only receive the last version you turned in. To see a list of the files you turned in, you may use the command "turnin -ls $cs352\_assg1$". For help information about turnin program, please use the command "turnin -h". If you still have questions about turnin, please either stop by TA's office hours or email TA's.

Your code should follow the instructions in the "C coding guidelines". In particular, pay attention to proper documentations)

We will put the sample executables in the class home dir at /home/cs352/fall03/sample_exec/assg2 on lectura. The purpose of these samples is to give you an idea how the expected solution look like (how it runs, how it gets input, how it format the output, etc). Please check your output format against the output generated by the sample executables with "diff" command. If you found any bugs with the sample executables or have questions on them, please contact us.

1. Create a C program called *Qremove* that reads a text from the standard input, and prints it to the standard output after all 'Q' letters have been removed. Note that the flow control commands in C are very similar to the ones in JAVA, so 'while()' so you should be familiar with (e.g.) the syntax to '*while*' command.
   Answer:

```
#include<stdio.h>

int main()
{
  char ch;

  //Get the charaters from the input one by one
```

```
  //If it is not a capital 'Q', print it in the output stream
  //Do this till the End of File (Input)

  while((ch = getchar()) != EOF)
    {
      if(ch != 'Q')
        {
          putchar(ch);
        }
    }

  return(1);
}
```

2. It is well known that the function e^x can be computed by evaluating the the sum

$$\sum_{i=0}^{\infty} x^i/i! \ .$$

   (a) Write a function called *Exponent*, whose prototype is
   float Exponent(float x, float eps)
   that receives a float x and a float *eps* (stands for ε) and compute the sum

$$\sum_{i=0}^{k} x^i/i!$$

   where k is the smallest integer such that $x^i/i!$ is no larger than *eps*. Pay attention to efficincy — try to minimize the number of arithmetic operations performed by the function.

   (b) Write a program called *ExponentCheck* that repeatedly asks the user to enter two numbers (x and *eps*), and prints the value returned by the function *Exponent(x,eps)* that you created for the previous question. The program should check that $0 < x < 1$ and $eps > 0$). The program should exit if the value 0 is enter to either x or *eps*. In this case the value 0 should be returned. If an illegal value is entered, the program should return 1.
   Answer:

```
#include<stdio.h>

#define OR ||
#define ERROR 1

// This function (approximately) computes e^x using tailor expansion.
```

```
float Exponent(float x, float eps){
    float sum = 1 ;
    float last_element = 1 ;
    int i;

    for( i = 1 ; last_element > eps  ; i++ ) {
        last_element *= (x/i) ;
        sum += last_element ;
    }

    return sum ;
}

int main(){
    float x, eps ;
    while(1){
        printf( ``Enter x and eps, separate by a blank ``);
        if ( 2 != scanf(``%f%f'', &x, &eps ) ) return ERROR ;

        // At this point we assume that both x and eps are legal floats.
        // Next we check their values.
        if (x < 0 OR x>=1 OR eps < 0 ) return ERROR ;
        if (x == 0 OR eps == 0 ) return 0 ; /* Exit code ok */
        printf(``The result value: %f \n'', Exponent( x , eps) )  ;
    }
    return 0;
}
```

3. Write a C program (named sum.c) that reads (by scanf()) a list of integers from the stdin and print (by printf()) to the stdout the number of integers, the minimum integer, the maximum integer and the sum of all the integers. The input integers are separated by spaces and/or newlines. A sample run of the program (where "sum" is the executable compiled from sum.c) is below:

   lec:xxx > sum
   1 2 3
   4 5 6 7 8
   8, 1, 8, 36
   lec:xxx >

   In the above example, 1 through 8 are the input numbers. "8, 1, 8, 36" are number of integers, min, max and sum respectively. You may assume the input are all valid integers of type "int". When there is no integer comming from

stdin at all, your program should print "0, N/A, N/A, 0".

The macros INT_MIN and INT_MAX in limits.h containing the maximum and minimum possible value of an integer might help.

Answer:

```
#include <limits.h>
#include <stdio.h>

int main()
{
    int sum = 0,
        num = 0;
    int n;
    int max = INT_MIN,
        min = INT_MAX;

    /* Read from stdin one integer at a time
     * When EOF, ret == -1 and we exit the loop.
     * When invalid integer, ret == 0 and we exit the loop.
     */
    while (scanf(``%d'', &n) == 1) {
        if (n < min)
            min = n;
        if (n > max)
            max = n;

        sum += n;
        num ++;
    }

    /* Print the result */
    if (num <= 0)
        printf(``%d, N/A, N/A, %d\n'',
               num, sum);
    else
        printf(``%d, %d, %d, %d\n'',
               num, min, max, sum);

    return 0;
}
```