CSc352 Summer03 Homework Assignment 2

Start: 06/16/03
Due: 06/23/03 (9:00pm)
Turnin ID: cs352_assg2

In this assignment, you need to turn in 2 C source program files **mitsubishi.c** and **numio.c** using the following command:

```
turnin cs352_assg2 mitsubishi.c numio.c
```

You need to compile both programs using **gcc** with **–Wall** option. In order to encourage high-quality codes, compile-time warning will result in points off, so you need to make sure your program can be compiled by "gcc –Wall" cleanly. For example, when compiling numio.c, use command:

```
gcc –Wall numio.c
```

If there are any syntax errors or warnings, get rid all of them by modifying the source code.

1. Text Art - Mitsubishi Logo:

Write a C program (named **mitsubishi.c**) that takes an integer **n** (of type **int**) from the **stdin** and print the Mitsubishi Logo (formed by "*") to the **stdout** according to size **n**. **n** can be read from the **stdin** by the statement:

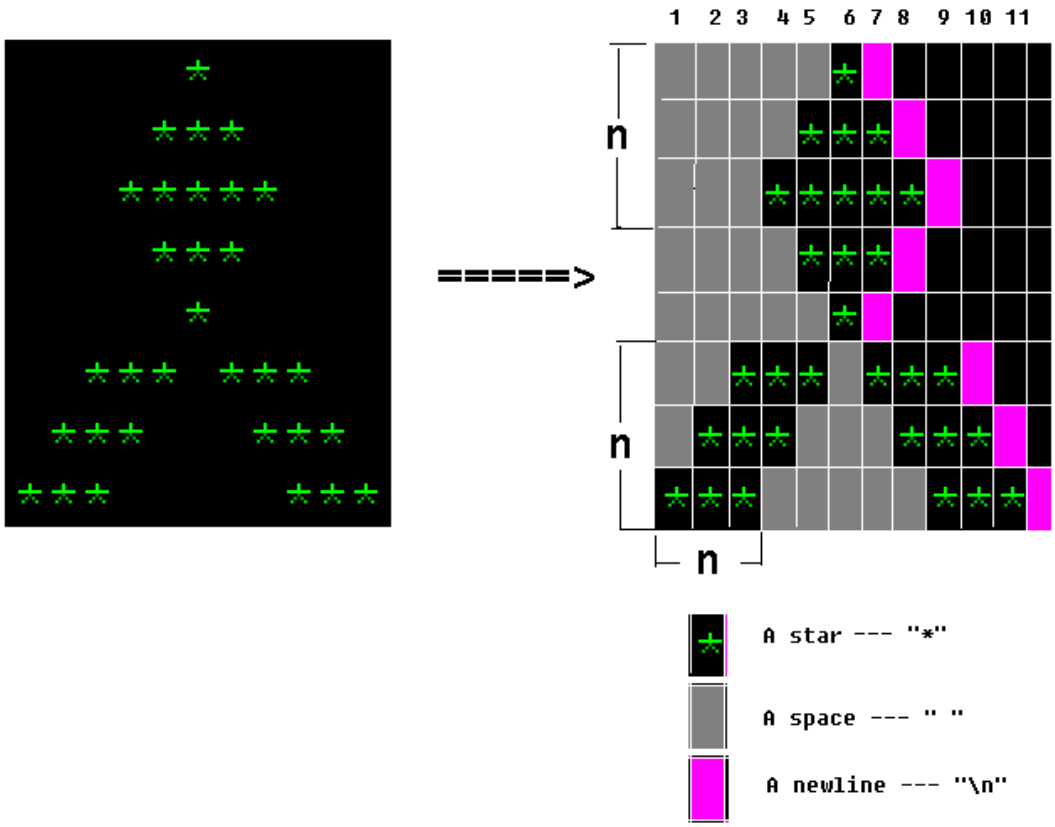
```
scanf("%d", &n);
```

You may assume the input **n** will always be an integer. However, right after reading the **n** from the **stdin**, you need to check if **n** is in the range [2, 30]. If not, give out an error message "Out of boundry" and then exit the program by the statement:

```
exit(1);
```

The definition of size **n** is shown in the following sample output. To make the grading easier, you should print **star** ("*"), **space** (" ") and **newline** ("\n") in the exact same way as the sample output shown below .

In the sample output below, the picture on the left is the actual output on the screen. The other colorful picture on the right is decorated by me to show you the exact characters and location to print. You don't need to print the grid and different colors!! Can you? ☺



From the picture on the right, you need to pay attention to the following things:

(1) The most left **star** ("*") of the logo must be printed in column one (I numbered the columns on the picture). In another word, the logo must touch the left side of the screen.
(2) Don't add blank lines on top or under bottom of the logo. If the logo consists of **m** lines of characters, you should only print **m newlines** (shown with pink cells in the picture), each of which should follow one line of the logo.

Anyway, an executable will be provided to you to get rid of any possible confusion in the write-up. Whenever you are not sure about something, you could always run the provided executable to see what is supposed to be in the result output.

In order to make sure your output is in the correct printing format. You may run your program and the provided executable with the same input **n**. Direct each result to a file. Then compare the 2 files. Your output should be exactly the same as the one produced by the provided executable. (You may want to use the tool **diff** to compare the 2 output files.)

2 High resolution calculations – Huge integer representation:

In C, even the integer type "long long int" can only handle numbers of size up to 8 bytes. Thus 2^64 = 18446744073709551615 is the biggest integer that can be represented by integer types in C. This number ONLY has 20 digits. What if we want to add two integers with 250 digits each? There are no integer types in C that we can use directly!

In this problem, you are to implement a C program named **numio.c** that can input, store and output huge integers of size at most **MAX_DIGITS** digits (here **MAX_DIGITS** is a constant defined by "#define **MAX_DIGITS 250**" in the provided header file **numio.h**). In later assignments, we will extend this program to do the +, -, * and / operations on those huge integers.

(1) We will use a char array to store a huge integer. Each element of the array holds one digit. The 0^{th} element (the one with subscript 0), contains the number of digits in the huge integer. The 1^{st} element contains the least significant digit of the huge integer. The 2^{nd} element contains the 2^{nd} least significant digit, and so on.

For example, denote the char array as **num**. An integer 23456 is stored in "char num[**MAX_DIGITS+1**]" as the following:

num[0] = 5 // number of digits in 23456
num[1] = 6 // the least significant digit
num[2] = 5
num[3] = 4
num[4] = 3
num[5] = 2

Why we define **num** as "char num[**MAX_DIGITS+1**]" instead of "char num[**MAX_DIGITS**]"? Think about it, but you don't have to answer it. However, if you have difficulty in answering it, please come to our office hours.

(2) You need to write a function to read a huge integer from the **stdin** and store it in a char array. The prototype of the function is:

```
int read_num(unsigned char *num);
```

num is the pointer to the char array.

Since huge integers will cause the overflow of C integer types, you need to read the huge integer from the **stdin** as a string (actually char arrary in C). Every element of the string is a character '0', '1', '2', …, '9'. You need to transform each of them into an integer 0, 1, 2, …, 9 and store them in the char array **num** in the format we discussed above in (1). (Notice that '1' is different from 1. When a char variable stores the character '1', its actually integer value is 49. Please refer to the ASCII table http://www.asciitable.com)

If the number of digits in the input huge integer is less than 1 or greater than **MAX_DIGITS**, immediately return an error code **CERR_LEN_ILLEGAL**. If any digit in the input is some character other than '0', '1', '2', …, '9', return error code **CERR_DIG_ILLEGAL**. If everything is correct, return **CERR_OK**, telling the

caller that the work is done correctly. All those error codes are also defined in a provided header file **numio.h**.

(3) You need to write a function to print a huge integer to the **stdout**. The prototype of the function is:

```
int print_num(unsigned char *num);
```

It should simply print the huge integer represented by the char array **num** to the **stdout** without any prefix or suffix like **space**, **newline**, etc. In this assignment, you can assume the argument **num** passed into this function is always correct (it is in the correct length range, all the digits stored in it is between 0 and 9, etc.). The later assignment, we will add more error handling.

For example, the huge integer "23456" will be printed just like:

```
23456
```

Yes, the output is exactly the same as the input. However, if your program just print out the input string without any transformation mentioned above, you may get nothing in your grade. An executable will be provided to help you to check your output in the way described in problem 1.

(4) Finally write a main() function, in which you need to the following things:
(a) Declare a char array to store huge integers;
(b) Call function `read_num()` to read an huge integer into the char array declared above;
(c) If the return value from `read_num()` is not CERR_OK, return 0;
(d) Call function `print_num()` to print the huge integer in the char array declared above;
(e) Print an ending **newline** with "printf("\n");"

(5) All the function prototypes and useful constants (e.g. the max number of digits allowed, all return codes, etc.) are defined in the header file **numio.h**. You are to copy numio.h to the same directory where your **numio.c** is stored. Then include it at the beginning of your **numio.c** file. Do NOT modify the **numio.h** file.

In **numio.c** you write, you need to implement at least 3 functions: `read_num()`, `print_num()` and `main()`. `read_num()`and `print_num()`must be defined according to the prototype give above (and also available in the given **numio.h** file). If you need to define other helper functions called in `read_num()`, `print_num()` and `main()`, please define them as static functions, which are only accessible in the file **numio.c**.

All the header files and executables are in the cs352 home directory
/home/cs352/summer03/assignments/hw2