CSc352 (Summer03) Homework 5

Start: 07/14/03
Due: 07/21/03
Turnin ID: cs352_assg5
Turnin file list: **hw5.tar.gz** (which contains all the programs and **Makefile**. Details of **hw5.tar.gz** will be discussed at the end of this write-up)

0. Background Readings

- gzip program
- gunzip program
- tar program
- ar program

1. Magic Cards

You have a pile of **n** ($1 \leq \mathbf{n} \leq 50$) cards. We write the numbers 1, 2, 3, …, n on each of the cards. No two cards have the same number. The cards are identified by the number written on them. Please print out the ordering of the cards that satisfies the following conditions:

(1) You do the following operations on the pile of cards:
   1. Move the top card to the bottom of the pile.
   2. Throw away the current top card from the pile.
   3. Repeat the above 2 steps until all the cards are thrown away from the pile and the pile is empty.

(2) After doing the above operations, you observe the order, in which you threw the cards. Surprisingly, the order you have thrown away cards is just 1, 2, 3, …, n

The following is an example. You have a pile of 3 cards. You print out, as the result of your program, the order of these cards in the order of 2-1-3, which means 2 is on the top of the pile, 1 is the 2nd card in the pile, and 3 is the bottom card of the pile. If you look at the operations (listed below) done to the pile, you will know this order 2-1-3 satisfies the requirement above.

(Originally) 2-1-3
(After moving the top card 2) 1-3-2
(After throwing the **card 1**) 3-2
(After moving the top card 3) 2-3
(After throwing the **card 2**) 3
(After moving the top card 3) 3
(After throwing the **card 3**) <Empty pile>

The number **n** is read from the **stdin**, the output result is just a list of the card numbers separated by "-" as seen above. If **n** is out of boundary, print the error message **"Wrong input"** to the **stderr**, and **exit(1)**. Otherwise, print out the result and **return(0)**.

2. Bitmap Library

Implement a library named **libbm.a**. The function of this library is specified in the header file **bitmap.h**.

- **int bm_init(int bits);**

  This function allocates a bitmap of **bits** bits and initializes all the bits to 0. This is a

0-based bitmap. So in this bitmap, we identify the bits as bit 0, bit 1, bit 2, …, bit (**bits**-1). If **bits**≤0 or **bits**≥**MAX**, return the error code **ERR_BOUNDARY**. If some memory related errors happened in this function (e.g. malloc() fails), return the error code **ERR_MEM**.

- **void bm_set(int n);**
  This function set bit **n** to 1. If there is no such bit **n** in the bitmap (e.g. bit -1, bit **bits**, where **bits** is the parameter we used when calling **bm_init** function), we do nothing in this function.
- **void bm_clr(int n);**
  This function set bit **n** to 0. If there is no such bit **n** in the bitmap (e.g. bit -1, bit **bits**, where **bits** is the parameter we used when calling **bm_init** function), we do nothing in this function.
- **int bm_get(int n);**
  This function returns the value of bit **n** (the return value should be either 0 or 1). If there is no such bit **n** in the bitmap (e.g. bit -1, bit **bits**, where **bits** is the parameter we used when calling **bm_init** function), we simply return 0.

In this library, you don't have to print out anything.

You must use one single bit to represent a bit of the bitmap. If the total number of bits in the bitmap is 32, you should NOT use more than 4 bytes to store the whole bitmap.

Because we will require you to write a **Makefile** for this assignment, you can choose whatever filename you like. When grading, we just run command "**make**" and with your **Makefile**, correct executables and libraries should be produced. For example, if you wrote 2 files for the bitmap library, namly file1.c file2.c.

First, you need to compile the C program you write into one or multiple object files.

```
gcc –c -Wall -I/home/cs352/summer03/assignments/hw5 file1.c file2.c
```

Then you need to use the **ar** program to make the library file **libbm.a** from those object files.

```
ar –rs libbm.a file1.o file2.o
```

Detailed requirements will be listed below.

3. Packaging

(1) You are to write a **Makefile** for this assignment.
(2) Put all the files you wrote into a directory named **hw5**.
(3) Tar and zip the directory hw5 with commands **tar** and **gzip**.

```
tar cvf hw5.tar hw5
gzip hw5.tar
```

(4) Turnin the **hw5.tar.gz** file.

We don't care about the filenames of your .c files. You can name your .c files whatever you like. When grading, we simply run the command "**make**" and 3 files should be produced: **card**, **libbm.a** and **test_bm**.

card is the executable for problem 1 (Magic cards).

libbm.a is the bitmap library.

test_bm is the executable of the provided bitmap testcase. You produce this executable by compiling **test_bm.c** (provided by us) and then linking **test_bm.o** with **libbm.a**.