CS 213, Spring 2000
Lab Assignment L1
Assigned: Jan. 20, Due: Thursday, Feb. 3, 00:01AM

Antonia Zhai (zhaia@cs.cmu.edu) is the lead person for this assignment.

The purpose of this assignment is to become familiar with bit-level representations. In this assignment, some of the requirements are artificial, but you will find yourself thinking much more about bits in working your way through them. As a side-effect, you will learn a reasonable amount about DES (Data Encryption Standard), which is one of the most commonly used schemes for encrypting messages.

Situation

You are trapped in the basement of the World Trade Center, with a bomb, which is controlled by a computer.

Luckily, there is a list of instructions that you can follow to deactivate the bomb. But the instructions are encoded using an algorithm called DEADS, which is a DES like encryption scheme. See the appendix for how DEADS works.

Also luckily for you, the key that is used to encrypt the message is also stored on the computer. The key has two parts: the lower half is `0xABFD9276`, the higher half is `0x9631DCCA`.

Since you are trapped, you have to decode the message on the machine that controls the bomb. Since you are using this strange computer that is used to control the bomb, there are a lot of constraints associated with the program you can write on this machine. We summarize them as rules under the programming rules section.

Logistics

You may work in a group of up to 2 people in solving the problems for this assignment. The only "hand-in" will be electronic. Any clarifications and revisions to the assignment will be posted on Web page `assigns.html` in the class WWW directory.

All files for this assignment are in the lab directory:

`/afs/cs.cmu.edu/academic/class/15213-s00/L1`

You are given the following files:

- Makefile;
- team.h, team.c;

- bits.c;
- decode.c;
- code1, code2, code3, code4

Your code will go in `bits.c` Take a look at the file `team.c`, you'll notice a C structure `team` into which you should insert the requested identifying information about the one or two individuals comprising your programming team. Do this right away so that you don't forget. To compile the decoder, use the command `gmake decode`. This command will create the executable program, `decode`.

The files code1 through code4 are the instructions to deactivate the bomb. You must decode each message. You also know the following: code1 is only encoded with the first round of DEADS encryption, code2 is encoded with the first two rounds of DEADS encryption, and so on. The output of your program should be the instructions to deactivate the bomb.

Programming Rules

- You are not allowed to declare your own functions, other than the ones given;
- You do not have to implement each and every function in `bits.c`. If you choose not to implement a certain function, have that function always return `0`. However, if you do choose to implement a certain function, you must implement it according to the specification written in the comment above the procedure.
- In `bits.c`, the following rules apply:
  - You are allowed to use integer constants `0x0` through `0xFFFFFFFF`, inclusive.
  - You are allowed to use unary operator `!`
  - You are allowed to use binary bit operators `&`, `|`, `^`, <<, and >>, binary arithmatic operators + and −.
  - You are allowed to read from and write to an array element.
  - You are allowed to make calls to any functions that are defined in the file.
  - You can use any control constructs such as `if`, `do`, `while`, `switch`, and `for`, but each use will deduct .5 point from your grade.
  - You are not allowed to use the unary bit operation `~`.
  - You are forbidden to use any other operator, including >, <, >=, <=, ==, `&&`, `||` and `?:`.
  - Everything else is forbidden in `bits.c`

Evaluation

30 Decoding the first round.

20 * 3 Each additional round decoded.

10 Subjective evaluation of the quality of your solutions and comments.

-.5 For each control structure

There is a limit on the total number of operators allowed, which will be posted on the bboard:

-0.2 points for each operator in `bits.c` that is over a certain threshold

+0.2 points for each operator below a certain threshold

Hand In

Make sure you have included your identifying information in your file `team.c`. Remove any extraneous print statements. Create a directory, using the Andrew Id of the first person on your team. Make a copy of the files `team.c bits.c` into the directory. Copy this directory to

```
/afs/cs.cmu.edu/academic/class/15213-s00/L1/handin
```

You only have write permission to this directory. If you make a mistake, e.g., by giving the file an incorrect name, send a copy of the correct file via email to the contact person for this assignment along with an explanation.

You may only hand in an assignment once. Make sure your code is in its final form before handing it in.