Marks : 12 **CSCI131**

Spring Session 2015

Due date: August 21st

Assignment 1
Bit manipulations
5 marks

Complete the associated exercises before attempting the assignment

Aim

This assignment introduces coding in the C language, utilising the NetBeans IDE, and provides practice with bit manipulations and binary, octal, and hexadecimal representations of bit patterns.

The assignment should be completed in the laboratory using the Ubuntu (Linux) environment.

Objectives

On completion of this assignment and its associated exercise, you will be able to:

- Use NetBeans to create a C application.
- Exploit the gdb debugger.
- Manipulate bit data using logical operations and shifts.
- Interconvert binary, octal, and hexadecimal representations of bit data.

Task

Typically, at this stage in a subject introducing computer systems, students are given tests involving long series of exercises to complete. Exercises such as:

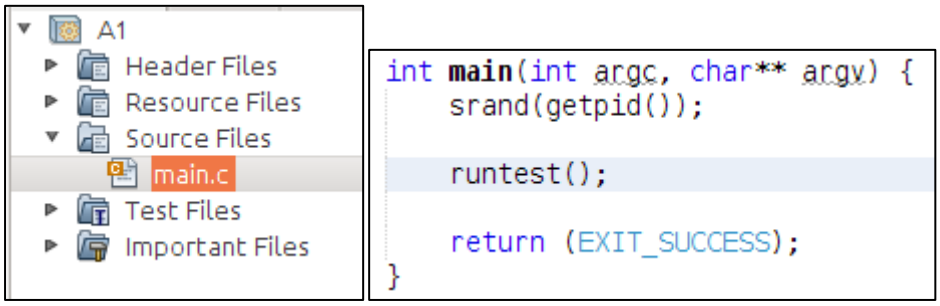
```
0x3c4   ^ 0x1a97b =
0x14a2  | 0xec451 =
0377    ^ 0105736 =
0100110101110101b << 2 =
```

Such exercises provide familiarity with logical and shift bit-manipulation operations, and practice with alternative representations of bit data.

You will have to complete such a test.

More importantly, you are writing a program that will create individual tests for students. You will test your own understanding of bits by taking the test that your program generates.

You will create a **C application** in NetBeans:



The program's runtest() method will present the user with a set of three tests:

- Logical operations – combine two bit values using &, |, ^, and negate bits using ~.
- Logical left and right shifts
- Arithmetic left and right shifts

Each test will present the user with a series of problems. Randomly chosen data values will be shown, a bit manipulation operation will be specified, and then the user will be required to enter a value for the result of that operation as performed on the data displayed. The value entered by the user will be compared with the correct result as computed by the program. If the user enters the correct value, the test sequence continues. If a value is incorrect, the tests start over with a new sequence of random data values. The user will be required to answer correctly twelve successive questions on logical operations, and six successive questions for each type of shift.

When displaying data, or requesting input, the program will chose at random to use binary, octal, or hex notation.

Example fragments of output from my version of the program are as shown below. For the "logical operations" test, my program prints the values for two randomly chosen unsigned short integers – variables "a" and "b"; the output format, binary/octal/hexadecimal is chosen at random. (When displaying octal and hexadecimal, the program outputs the values with a leading 0 for octal, and a 0x for hexadecimal so as to avoid any ambiguities.) The program then selects at random one of a set of possible logical formulae:

- ~a
- a & b
- a & ~b
- a | b
- a ^ b
- ~(~a & ~b)

The formula is shown. The user is then required to enter the correct value for that formula as applied to the data shown. The format used for input is again chosen randomly.

```
a = 0100001010001010b
b = 01110000001010llb
~a
Enter data in hexadecimal : bd75
Correct
a = 1000111110001010b
b = 0001111100000000b
a ^ b
Enter data in binary      : bits :1001000010001010
Correct
a = 023640
b = 072415
a & ~b
Enter data in hexadecimal : 2a0
Correct
a = 0x22c2
b = 0x8ec2
a & ~b
Enter data in octal       : 20000
Correct
a = 0000011111111110b
b = 1001000001010010b
~a
Enter data in binary      : bits :11111000000001
Incorrect, start all over again
a = 0xe46
b = 0x9baf
a & ~b
Enter data in hexadecimal : 440
Correct
a = 0x8d36
b = 0xc909
a | b
```

The tests using shifts follow a similar pattern. The program randomly picks a short integer value (unsigned short for logical shifts, cast to signed short for arithmetic shifts), a direction (left or right), and a number of places by which to shift (1..3). It displays details of the chosen combination (randomly choosing binary/octal/hexadecimal format for the value) and gets the user to enter a value for the result of the operation.

You can remind yourself about bit shift operations at
http://en.wikipedia.org/wiki/Bitwise_operations_in_C

Shift operators [edit]

There are two bitwise shift operators. They are:

- Right shift (>>)
- Left shift (<<)

Right shift >> [edit]

The symbol of right shift operator is **>>**. For its operation, it requires two operands. It shifts each bit in its left operand to the right. The number following the operator decides the number of places the bits are shifted (i.e. the right operand). Thus by doing ch >> 3 all the bits will be shifted to the right by three places and so on.

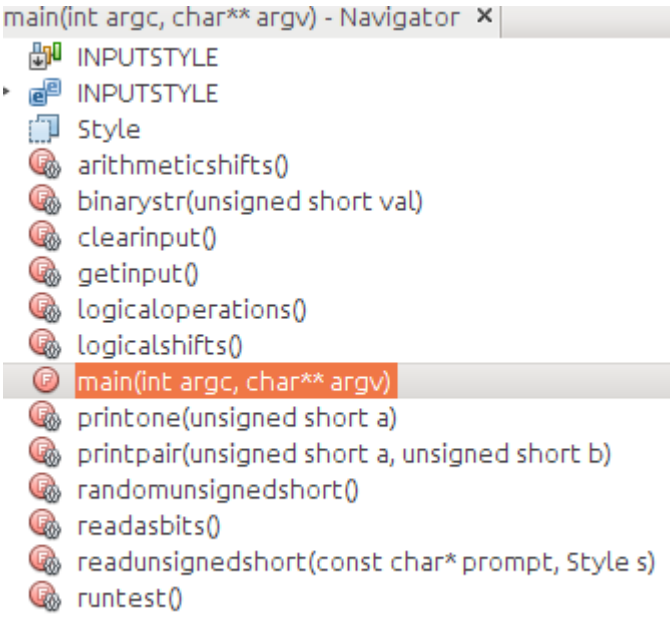
Example:

If the variable *ch* contains the bit pattern *11100101*, then *ch >> 1* will produce the result *01110010*, and *ch >> 2* will produce *00111001*.

Here blank spaces are generated simultaneously on the left when the bits are shifted to the right. When performed on an unsigned type, the operation performed is a logical shift, causing the blanks to be filled by 0s (zeros). When performed on a signed type, an arithmetic shift is performed, causing the blank to be filled with the sign bit of the left operand.

```
Logical shift operations
You must get 6 successive questions correct
a = 0174405
a << 3
Enter data in octal       : 144050
Correct
a = 1101010110010111b
a >> 3
Enter data in binary      : bits :1101010110010
Correct
a =0xc239
a << 2
Enter data in hexadecimal : 8e4
Correct
a = 0100100000101101b
a << 1
Enter data in binary      : bits :1001000001011010
Correct
a =0xc169
a >> 3
Enter data in hexadecimal : 182d
Correct
a = 1001100010000110b
a >> 1
Enter data in octal       : 46103
Correct
Arithmetic shift operations
You must get 6 successive questions correct
a =0xdc2c
a << 2
Enter data in hexadecimal : 70b0
Correct
a = 013610
a << 2
```

Much of the code for this application can be "salvaged" from the associated exercises. My program's structure was:



Assignment report

You do not submit your code files!

You are marked on a report that you write. This report presents details of your NetBeans project, your code, and evidence for correct operation in the form of a record of your completion of a test as administered by your program. The report should be prepared on a word processor and converted to PDF format. The Open Office word processor on Ubuntu Linux has a "print to PDF" option. There should be Word to PDF converters on the Windows OS. Ubuntu has a snapshot tool (in its accessories menu).

Do **not** use screen shots for your code. NetBeans has a "Print to HTML" option for code files. Use this to create files with formatted code (keywords in different font etc). The "Print to HTML" option allows you to include or omit line numbers; it is better to omit them. The formatted code in the generated HTML file can then be copy-pasted into your report. (When code is included in this way, markers can easily search your code when checking your submission.)

Submission

Prepare your report and convert to PDF format as the file A1.pdf.

Submission is done electronically via a program called `turnin` that runs on "banshee" – the main CS undergraduate machine. You must first transfer your A1.pdf file to your home directory on banshee (this is different from the home directory that you access on the Linux machines). You can transfer the file using a SSH file-transfer program. The Ubuntu OS allows you to open a file-browser connected to your banshee home directory – and you can simply drag your A1.pdf file across using the visual file browser.

For CSCI131, assignments are submitted electronically via the `turnin` system. For this assignment you submit your assignment via the command:

```
turnin -c csci131 -a 1 A1.pdf
```

Late submissions would be submitted as:

```
turnin -c csci131 -a 1late A1.pdf
```

The program `turnin` only works when you are logged in to the main banshee undergraduate server machine. From an Ubuntu workstation in the lab, you must open a terminal session on the local machine, and then login to banshee via ssh and run the `turnin` program.

Marking

The assignment is worth 5 marks total.

- Appearance and structure of report: 1 mark
- C code: 3 marks
- Record of completion of test administered by program: 1 mark