Problem Set 1

Topics: Writing, compiling, and debugging programs. Preprocessor macros. C file structure. Variables. Functions and program statements. Returning from functions.

Problem 1.1

(a) What do curly braces denote in C? Why does it make sense to use curly braces to surround the body of a function?

(b) Describe the difference between the literal values 7, "7", '7'.

(c) Consider the statement

```
double ans = 10.0+2.0/3.0-2.0*2.0;
```

Rewrite this statement, inserting parentheses to ensure that `ans = 11.0` upon evaluation of this statement.

Answers:

(a) Curly braces defines a code block or the body of a function, which means a region of code. Because in literature or math we also use curly braces to include some content or information.

(b) ~~The type for 7 is integer, and type for "7" is string while type for '7' is char.~~

The first literal describes an integer of 7. The second describes a **null-terminated** string consisting of the character '7'. The third describes the character '7', whose value is its ASCII character code(55).

(c)

```
double ans = 10.0+2.0/((3.0-2.0)*2.0);
```

Problem 1.2

Consider the statement

```
double ans = 18.0/squared(2+1);
```

For each of the four versions of the function macro **squared()** below, write the corresponding value of `ans`.

1. `#define squared(x) x*x`
2. `#define squared(x) (x*x)`
3. `#define squared(x) (x)*(x)`
4. `#define squared(x) ((x)*(x))`

Answers:

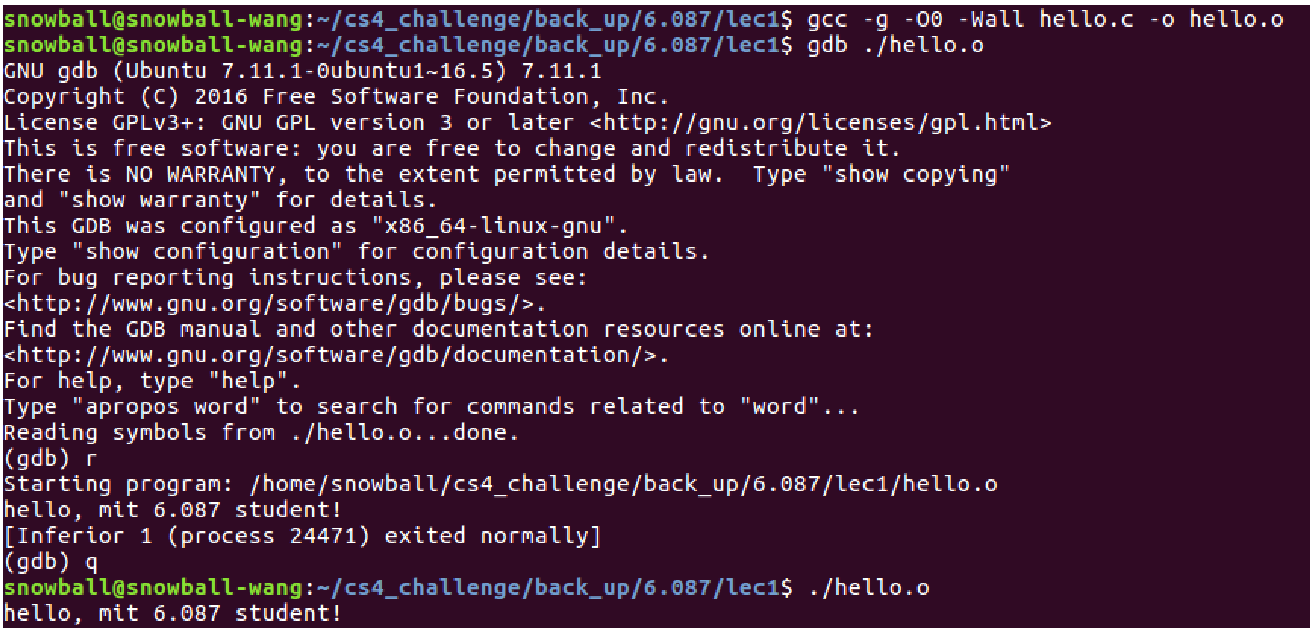
1. `ans = 12.0`
2. `ans = 3.6`
3. ~~`ans = 27.0`~~ `ans = 18.0/(2+1)*(2+1)`,so `ans = 18.0` **DON'T be sloppy!**
4. `ans = 2.0`

Problem 1.3

Write the "Hello, 6.087 students" program described in lecture in your favorite text editor and compile and execute it. Turn in a printout or screen shot showing

- the command used to compile your program
- the command used to execute your program(using `gdb`)
- the output of your program

Answers:



```
snowball@snowball-wang:~/cs4_challenge/back_up/6.087/lec1$ gcc -g -O0 -Wall hello.c -o hello.o
snowball@snowball-wang:~/cs4_challenge/back_up/6.087/lec1$ gdb ./hello.o
GNU gdb (Ubuntu 7.11.1-0ubuntu1~16.5) 7.11.1
Copyright (C) 2016 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from ./hello.o...done.
(gdb) r
Starting program: /home/snowball/cs4_challenge/back_up/6.087/lec1/hello.o
hello, mit 6.087 student!
[Inferior 1 (process 24471) exited normally]
(gdb) q
snowball@snowball-wang:~/cs4_challenge/back_up/6.087/lec1$ ./hello.o
hello, mit 6.087 student!
```

Problem 1.4

The following lines of code, when arranged in the proper sequence, output the simple message "All your base are belong to us."

1. `return 0;`
2. `const char msg[] = MSG1;`
3. `}`
4. `#define MSG1 "All your base are belong to us!"`
5. `int main(void){`
6. `#include <stdio.h>`
7. `puts(msg);`

Write out the proper arrangement(line numbers are sufficient) of this code.

Answers:

6 4 5 2 7 1 3

Problem 1.5

For each of the following statements, explain why it is not correct, and fix it.

(a)

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

(b)

```
int function(void arg1)
{
    return arg1-1;
}
```

(c)

```
#define MESSAGE = "Happy new year!"
puts(MESSAGE);
```

Answers:

(a) The header file doesn't need to add semicolon at the end of the statement.

correct:

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

(b) ~~The parameter for the function is void, which is not allowed. And there are no variable named `arg1-1` inside the function, so the `return arg1-1` is invalid.~~

The `void` type signifies an empty variable, which can not be used anyway.

correct:

```
int function(int arg1)
{
    return arg1-1;
}
```

(c) ~~Every c program takes `main()` function as the entry for the program. We cannot print out the `MESSAGE` without having a `main()` function.~~

The form of `#define` statement does not include an assignment operator.

correct:

```
#define MESSAGE "Happy new year!"
puts(MESSAGE);
```