ENEE 140 Lab 6

Lab instructions

This handout includes instructions for the recitation sessions on Wednesday and Friday. **Follow these instructions** to review the C arithmetic operations, then **submit the homework** as indicated below. To prepare for the next lecture, complete the **reading assignment** and try to solve the **weekly challenge**.

1 Basic arithmetic expressions

Question 1: Think about what will be the value of the following operations. Then write a simple program to check your answer. Your program will ask for two integers and output the value of (first_number) % (second_number).

```
-7 % 3 = ?  7 % -3 = ?  -7 % -3 = ?
```

Question 2: Insert ()'s in the following expression to indicate the precedence and give the result

```
5   %   3   * 2   +   12   /   5   -   -   3.6   /   9   =
```

Homework

Due: March 4 at 11:59 pm.

Create two programs and a text file by following the instructions below. Submit them using the following commands:

```
submit 2016 spring enee 140 AAAA 6 type_conversions.txt
submit 2016 spring enee 140 AAAA 6 toupper_function.c
submit 2016 spring enee 140 AAAA 6 clock_coordinates.c
```

Note: you must replace `AAA` with your own section number (0101, 0102, etc.)

1 Explicit and implicit casts

Write your answer to the questions below in a plain text file and name it `type_conversions.txt`. You do not have to copy your code into this file.

1. Answer the following questions and try to remember the correct answers, which you can obtain by writing a small program.

```
sizeof(int)    =                          sizeof(float)      =
sizeof(double) =                          sizeof(char)       =
sizeof (167)   =                          sizeof(3.1415926)  =
sizeof('$')    =                          sizeof("ENEE140")  =
```

2. For the following expressions, identify the implicit cast and explicit cast. Answer for the first one is given to you as an example:

```
int a = 2, b = 3;
float f = 2.5;
double d = -1.2;
int int_result;
float real_result;

int_result = a * f;
// a is casted implicitly to float by the multiplication operation a*f,
// the product is then casted implicitly to int by the = (assignment) operation.
real_result = a * f;
real_result = (float) a * b;
d = a + b / a * f;
d = f * b / a + a;
```

3. In C, when an integer is divided by another integer, the result will also be an integer. What happens when there is a remainder like in 23/5? Will the result be 4, 5, 4.6, or something else? Write a program to find out the rule (by testing a few examples). Suggested answers (only one is correct):
   (a) round to the closest integer
   (b) truncate the integer part
   (c) round down
   (d) round up

4. In C, the explicit cast operation can change a variable from one data type to another. When we change one from low precision/accuracy to high (for example from float to double), we will not lose any information. What happens if we cast the other way, for example, when casting a float variable to int? As for the previous question, you should write a program to help you identify the rule. Suggested answers (only one is correct):
   (a) round to the closest integer
   (b) truncate the integer part
   (c) round down
   (d) round up

2 Refactoring

Rewrite the sum of squares program you wrote for the previous homework by implementing a function called `sum_of_squares`, which is invoked in the following main function. You can do this by moving the relevant code from your old program to the body of this new functions (this operation is called "refactoring"). Name the new program `sum_of_squares_function.c`. Your program should produce output identical to the original program.

```
#include <stdio.h>

int
main()
{
    long ss, n;

    // Read in a character
    printf("Input an integer: ");
    scanf ("%ld", &n);

    // Compute the sum of squares
    ss = sum_of_squares(n);    // <-- Implement this function

    // Print the result
    printf ("The sum of squares is %ld\n", ss);

    return 0;
}
```

3 Clock coordinates

Write a program called `clock_coordinates.c` that prompts the user for the radius of an analog clock, and then prints the x and y coordinates of the 12 hour marks for that clock. You may assume that the origin for the x and y axes is in the center of the clock.

Hint. You will have to include the `math.h` header and use the `sin()` and `cos()` functions to compute the coordinates. Remember that these functions take in angles expressed in radians, so the constant `M_PI`, which stores the value of π and is also defined in `math.h`, may come in handy. Think about the angle that corresponds to each hour mark, from 1 to 12, and how to derive the coordinates from this angle.

Reading assignment

K&R Chapters 1.6, 1.9, 2.3, 2.4, 4.1, 4.2, B3.

Weekly challenge

Implement a function that copies one string into another and returns the number of characters copied.

You can use the following template (also available in the GRACE class public directory, at `public/challenges/week06`):

```
/*
 * strncpy.c
 *
 *  Copy one string to another.
 *  Correctly terminate the new string and do not exceed its bounds.
 */


#include <stdio.h>


/* Copy src into dst; return the number of characters copied */
int
strncpy(char dst[], const char src[], int dst_size)
{
}


int
main()
{
        char from[20] = "";
        char to[10];
        int char_copied;

        printf("Enter a string: ");
        scanf("%19s", from);

        char_copied = strncpy(to, from, sizeof(to));

        printf("%d characters copied; the destination string is: %s\n",
                        char_copied, to);

        return 0;
}
```

The weekly challenge will not be graded. However, if you manage to solve it, you may submit it for extra credit. The deadline for submitting your solution to the weekly challenge is **Monday at 11:59 pm**. To be eligible for extra credit, you must implement correctly **all but two** of the weekly challenges. You can submit your program from a GRACE machine using the following command (replace `AAAA` with your section number):

```
submit 2016 spring enee 140 AAAA 1006 strncpy.c
```