Course: Programming Fundamentals – **ENCM 339**
Lab #: Lab 2
Instructor: S. Norman
Student Name: **Mitchell Sawatzky**
Lab Section: **B02**
Date Submitted: **Sept 22, 2015**

Exercise C

h-min-s.c

```c
//  h-min-s.c

//  Started by:   Steve Norman
//
//  Completed by: Mitchell Sawatzky
//   Lab section: B02
//          Date: Tuesday Sept 22, 2015

#include <stdio.h>
#include <stdlib.h>

#define SEC_PER_HOUR 3600
#define SEC_PER_MIN 60

void s2hms(int total_s, int *h, int *min, int *s);
//
// Converts a number of seconds into hours, minutes, and seconds.
// For example, converts 3678s into 1h, 1min, 18s.
// REQUIRES:
//   total_s >= 0. Pointer arguments all point to appropriate variables.
// PROMISES:
//   Hour result is in *h.
//   Minutes result is in *min.
//   Seconds result is in *s.

int main(void)
{
  int s_in, h_out, min_out, s_out, scan_count;

  printf("Enter a time interval as an integer number of seconds: ");
  scan_count = scanf("%d", &s_in);
  if (scan_count != 1) {
    printf("Unable to convert your input to an int.\n");
    exit(1);
  }
  if (s_in < 0) {
    printf("I can't work with a negative number of seconds!.\n");
    exit(1);
  }
  printf("Doing conversion for input of %d seconds ... \n", s_in);

  // MAKE A CALL TO s2hms HERE.
  s2hms(s_in, &h_out, &min_out, &s_out);

  printf("That is equivalent to %d hours(s), %d minutes(s), %d second(s).\n",
   h_out, min_out, s_out);

  return 0;
}

// WRITE A DEFINITION FOR s2hms HERE.
// Hints:
//    1. Use SEC_PER_HOUR and SEC_PER_MIN as defined above.
//    2. / for integer division and % for integer remainder are useful here.

void s2hms(int total_s, int *h, int *min, int *s)
```

```
{
    *h = total_s / SEC_PER_HOUR;
    *min = total_s % SEC_PER_HOUR / SEC_PER_MIN;
    *s = total_s % SEC_PER_HOUR % SEC_PER_MIN;
    return;
}
```

Terminal Output

```
Mitchell@ttys001 00:13 {0} [lab2]$ ./test.out
Enter a time interval as an integer number of seconds: 0
Doing conversion for input of 0 seconds ...
That is equivalent to 0 hours(s), 0 minutes(s), 0 second(s).
Mitchell@ttys001 00:16 {0} [lab2]$ ./test.out
Enter a time interval as an integer number of seconds: 59
Doing conversion for input of 59 seconds ...
That is equivalent to 0 hours(s), 0 minutes(s), 59 second(s).
Mitchell@ttys001 00:16 {0} [lab2]$ ./test.out
Enter a time interval as an integer number of seconds: 63
Doing conversion for input of 63 seconds ...
That is equivalent to 0 hours(s), 1 minutes(s), 3 second(s).
Mitchell@ttys001 00:17 {0} [lab2]$ ./test.out
Enter a time interval as an integer number of seconds: 3599
Doing conversion for input of 3599 seconds ...
That is equivalent to 0 hours(s), 59 minutes(s), 59 second(s).
Mitchell@ttys001 00:17 {0} [lab2]$ ./test.out
Enter a time interval as an integer number of seconds: 3745
Doing conversion for input of 3745 seconds ...
That is equivalent to 1 hours(s), 2 minutes(s), 25 second(s).
Mitchell@ttys001 00:17 {0} [lab2]$ ./test.out
Enter a time interval as an integer number of seconds: 9013
Doing conversion for input of 9013 seconds ...
That is equivalent to 2 hours(s), 30 minutes(s), 13 second(s).
```

Exercise D

1.) Yes, both avg1.c and avg2.c behaved the same with valid inputs.
2.) No, avg1.c enters and infinite loop when it encounters a bad input, while avg2.c exits and prints an error message. This occurs because when scanf encounters an invalid input, it does not remove it from the input stream. avg1.c will continue to try to consume the invalid input, while avg2.c will exit when it detects that scanf could not read the input properly.

Exercise G

lab2exG.c

```
/* * * * * * * * * * * * * * * * * * * * * * * * *
 * Title: lab2exF.c                              *
 * Name: Mitchell Sawatzky                       *
 * UCID: 10146721                                *
 * Class: ENCM 339-T01/B02                       *
 * * * * * * * * * * * * * * * * * * * * * * * * */

#include <stdio.h>
#include <stdlib.h>

void get_input(double *km_in, int *hours_in, int *minutes_in);
// REQUIRES:
```

```
//     All three arguments as addresses of appropriate variables
// PROMISES:
//     Function will prompt for user inputs and fill each argument
//     with the appropriate values.

int get_int_or_die(void);
// REQUIRES:
//    User has been prompted to enter an int.
// PROMISES:
//    Function tries to read an int using scanf and "%d"
//    On success, that int is echoed to the user,
//    and the int is the function return value.
//    On failure, and error message is printed and
//    exit is called with an argument of 1.

double get_double_or_die(void);
// Like get_int_or_die, but tries to read a double using "%lf".

int main(void)
{
    int hours, mins, total_seconds;
    double dist;

    get_input(&dist, &hours, &mins);

    if (hours <= 0) {
        printf("Cannot compute a 0 or negative time!\n");
        exit(1);
    }
    if (mins <= 0) {
        printf("Cannot compute a 0 or negative time!\n");
        exit(1);
    }

    printf("Distance travelled: %lf km.\n", dist);
    printf("Time of travel: %d hour(s), %d minute(s).\n", hours, mins);
    printf("Average speed, in different units ...\n");

    total_seconds = 3600*hours + 60*mins;
    printf("\t%lf km/h.\n", (dist * 3600.0) / total_seconds);
    printf("\t%lf m/s.\n", (dist * 1000.0) / total_seconds);
    printf("\t%lf mph.\n", (dist * 1000.0 * 3600.0) / (5280.0 * 0.3048 * total_seconds));

    return 0;
}

void get_input(double *km_in, int *hours_in, int *minutes_in)
{
    printf("Please enter a distance in km, using type double.\n");
    *km_in = get_double_or_die();
    printf("Please enter number of hours, using type int.\n");
    *hours_in = get_int_or_die();
    printf("Please enter a number of minutes, using type int.\n");
    *minutes_in = get_int_or_die();

    return;
}

int get_int_or_die(void)
{
  int result;
  if (1 != scanf("%d", &result)) {
    printf("I could not read an int. I am quitting.\n");
    exit(1);
  }
  printf("I read an int value of %d.\n", result);
  return result;
}
```

```
double get_double_or_die(void)
{
    double result;
    if (1 != scanf("%lf", &result)) {
        printf("I could not read a double. I am quitting.\n");
        exit(1);
    }
    printf("I read a double value of %lf.\n", result);
    return result;
}
```

Terminal Output

```
Mitchell@ttys001 01:16 {0} [lab2]$ ./test.out
Please enter a distance in km, using type double.
297.2
I read a double value of 297.200000.
Please enter number of hours, using type int.
3
I read an int value of 3.
Please enter a number of minutes, using type int.
17
I read an int value of 17.
Distance travelled: 297.200000 km.
Time of travel: 3 hour(s), 17 minute(s).
Average speed, in different units ...
  90.517766 km/h.
  25.143824 m/s.
  56.245132 mph.
Mitchell@ttys001 01:16 {0} [lab2]$ ./test.out
Please enter a distance in km, using type double.
157.7
I read a double value of 157.700000.
Please enter number of hours, using type int.
1
I read an int value of 1.
Please enter a number of minutes, using type int.
24
I read an int value of 24.
Distance travelled: 157.700000 km.
Time of travel: 1 hour(s), 24 minute(s).
Average speed, in different units ...
  112.642857 km/h.
  31.289683 m/s.
  69.993026 mph.
Mitchell@ttys001 01:16 {0} [lab2]$ ./test.out
Please enter a distance in km, using type double.
No.
I could not read a double. I am quitting.
Mitchell@ttys001 01:17 {1} [lab2]$ ./test.out
Please enter a distance in km, using type double.
70
I read a double value of 70.000000.
Please enter number of hours, using type int.
7.8
I read an int value of 7.
Please enter a number of minutes, using type int.
I could not read an int. I am quitting.
Mitchell@ttys001 01:17 {1} [lab2]$ ./test.out
Please enter a distance in km, using type double.
70
I read a double value of 70.000000.
Please enter number of hours, using type int.
7
I read an int value of 7.
Please enter a number of minutes, using type int.
-90
I read an int value of -90.
Cannot compute a 0 or negative time!
```