UNIVERSITY OF WINDSOR
COMPUTER SCIENCE 60-140-01
MIDTERM TEST SOLUTION

Examiners: Dr. C.I. Ezeife
Given: Fri., Nov. 4, 2011

Student Name: ________________________________

Student Number: ________________________________

Lab. Section(circle one): 51, 52, 53, 54, other _________

________________________________________________________________________

INSTRUCTIONS (Please Read Carefully)

Examination Period is 1 hour 30 minutes. DO NOT WRITE WITH PENCIL.
Answer all questions. Write your answers in the spaces provided in the question paper.
May use only simple calculators. WRITE WITH PEN.
Total Marks = 100. Total number of questions = 3. Total number of pages = 7.

CONFIDENTIALITY AGREEMENT & STATEMENT OF HONESTY

I confirm that I will keep the content of this assignment/examination confidential. I confirm
that I have not received any unauthorized assistance in preparing for or doing this
assignment/examination. I confirm knowing that a mark of 0 may be assigned for copied
work.

________________________________________    ___________________________________
Student Signature                      Student Name (please print)

________________________________________    __________________________
Student I.D. Number              Date

For Marking Purposes Only (this part not to be filled by students)

<table>
<thead>
<tr>
<th>Question</th>
<th>Mark</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 (35 marks)</td>
<td></td>
</tr>
<tr>
<td>2 (15 marks)</td>
<td></td>
</tr>
<tr>
<td>3a (20 marks)</td>
<td></td>
</tr>
<tr>
<td>3b (10 marks)</td>
<td></td>
</tr>
<tr>
<td>3c (20 marks)</td>
<td></td>
</tr>
<tr>
<td>Total Mark (100 marks)</td>
<td></td>
</tr>
</tbody>
</table>
Problem 1:

1. Given the time conversion scales of:

1 hour = 3600 seconds, 1 hour = 60 minutes and 1 minute = 60 seconds.
1 second = 1/3600 = 0.00028 hour and 1 second = 1/60 = 0.017 minute,

Write a C program (using only top-down design approach with functions
Hr_to_MinSec and Sec_to_HrMin making only parameter calls, as given in the
structure chart below). You are required to write a C program for converting every
given input seconds (integer) to its hours, minutes, seconds (all integers) equivalent and
every given input hour (real) to its equivalent minute and seconds (all real), using the
two functions called by main function, as in the structure chart below. Your program
should read two input seconds and convert them to hours_minutes_seconds and should
also read two input hours and convert them to minutes, and also to seconds. Your
program should print the results in a conversion table showing the input values also, as
shown in the sample output.  ................(35 marks)

DO NOT USE GLOBAL VARIABLES. You should make only parameter calls (it is
up to you what type of parameter you should use to write your program), and read all
input data as well as print all desired output data in the main function. All conversion
operations should be done in the appropriate function. The function Hr_to_MinSec is
used to convert every given input hours (real) to minutes, and seconds (real), while the
function Sec_to_HrMin is used to convert every given second (integer) to
hours_minutes_seconds (integers).

Sample Input:
Please type the two seconds to be converted to hours, mins, secs.
3756  245
Please type the two hours to be converted to minutes and seconds:
3.40   1.23

Sample Output:
The Time Conversions from Seconds to Hours, Mins and Secs are

<table>
<thead>
<tr>
<th>Seconds</th>
<th>Hours, Minutes and Seconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>3756</td>
<td>1, 2, and 36</td>
</tr>
<tr>
<td>245</td>
<td>0, 4, and 5</td>
</tr>
</tbody>
</table>

*****
The Time Conversions from Hours to Minutes and Seconds are

<table>
<thead>
<tr>
<th>Hours</th>
<th>Minutes and Seconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.40</td>
<td>204.00 and 12240.00</td>
</tr>
<tr>
<td>1.23</td>
<td>73.80 and 4428.00</td>
</tr>
</tbody>
</table>

******* (note that this line is not part of output of programs *******

2
Use the following structure chart for solving the problem.

```
Control_Module
  0000

Sec_to_HrMin
  1000

Hr_to_MinSec
  2000
```

**SOLUTION**

/*
Problem: Given the time conversion scales of:
1 hour = 3600 seconds, 1 hour = 60 minutes and 1 minute = 60 seconds.
1 second = 1/3600 = 0.00028 hour and 1 second = 1/60 = 0.017 minute,
Write a C program (using only top-down design approach with functions
Hr_to_MinSec and Sec_to_HrMin making only parameter calls, as given in
the structure chart below). You are required to write a C program for
converting every given input seconds (integer) to its hours, minutes,
seconds (all integers) equivalent and every given input hour (real) to
its equivalent minute and seconds (all real), using the two functions
called by main function, as in the structure chart below. Your program
should read two input seconds and convert them to hour-minute-seconds
and should also read two input hours and convert them to minutes and
seconds. Your program should print the results in a conversion table
showing the input values also, as shown in the sample output.

The problem solution prints lines of output in a neat tabular form as shown
in the sample output, by calling the functions and without using
repetition
instructions.
*/

#include <stdio.h>

/* Three function prototypes                             */
void Sec_to_HrMin(int, int *, int *, int *);
void Hr_to_MinSec(float, float *, float *);

/* This is the main Module 0000 which calls modules Sec_to_HrMin and
Hr_to_MinSec                        */
int main(void)
{
    float Hrinput1, Hrinput2;
    int Secinput_a, Secinput_b;
    float Minout1, Secout1, Minout2, Secout2;
    int Hrout_a, Minout_a, Secout_a, Hrout_b, Minout_b, Secout_b;
```
/* Now the sequence of instructions */
printf("Please type the two seconds s to be converted to Hours, Minutes, Seconds:\n");
scanf("%d %d", & Secinput_a, & Secinput_b);
printf("Please type the two Hours to be converted to Minutes and Seconds:\n");
scanf("%f %f", &Hrinput1, &Hrinput2);

Sec_to_HrMin(Secinput_a, &Hrout_a, &Minout_a, &Secout_a);
Sec_to_HrMin(Secinput_b, &Hrout_b, &Minout_b, &Secout_b);

Hr_to_MinSec(Hrinput1, &Minout1, &Secout1);
Hr_to_MinSec(Hrinput2, &Minout2, &Secout2);

printf("The Time Conversions from Seconds to Hours, Mins and Secs are \n");
printf("\t Seconds \t | Hours, Minutes and Seconds \n");
printf("\t***********************************\n");
printf("\t %d \t	 | %d,  %d, and %d  
", Secinput_a, Hrout_a, Minout_a, Secout_a);
printf("\t %d \t	 | %d,  %d, and %d  
", Secinput_b, Hrout_b, Minout_b, Secout_b);

*****

printf("The Time Conversions from Hours to Minutes and Seconds are \n");
printf("\t Hours \t\t | Minutes and Seconds \n");
printf("\t***********************************\n");
printf("\t %0.2f \t\t | %0.2f  and %0.2f \n", Hrinput1, Minout1, Secout1);
printf("\t %0.2f \t\t | %0.2f  and %0.2f \n", Hrinput2, Minout2, Secout2);

return 0;
}

***** The function definition for function Sec_to_HrMin which accepts a Time in Seconds and prints its equivalent in Hours, Minutes and Seconds. This is called module 1000

*/
void Sec_to_HrMin(int Secinput, int *Hrout, int *Minout, int *Secout) {

int secondleft;

*Hrout = Secinput / 3600;
secondleft = Secinput % 3600;

*Minout = secondleft / 60;
*Secout = secondleft % 60;
}

***** The function definition for function Hr_to_MinSec which accepts a time in Hour and prints its equivalent in Minutes and Seconds. This is called module 2000
void Hr_to_MinSec(float Hrinput, float *Minout, float *Secondout) {
    *Minout = Hrinput * 60.0;
    *Secondout = Hrinput * 3600;
} /* Hr_to_MinSec ends*/

Marking Scheme:
Assign - 5 marks for proper definition of input and output variables/consts
- 5 marks for proper declaration of function prototypes
- 15 marks for correct logic in the control module body including correct use of program instructions and function calls.
(broken down as 5 marks for correct reading of input data, 5 marks for correct function calls, 5 marks for correct printing of output data and messages)
- 10 marks for function definitions (for each function, 2 marks for correct logic and 3 marks for correct function header and parameter passing)
***** the above will make up 35 marks and marks are lost for the following:
- 10 marks off for improper parameter passing mechanisms.
- Take 15 marks off if parameters are not used in the solution. Then, mark the solution based on 2 marks for variable declaration, 3 for function prototypes, 5 marks for control module body, 5 marks for function definition and 5 marks for correct printing of output.
- Take 20 marks off for not using functions at all. Then, mark the solution based on 5 marks for variable declaration, 5 marks for control module body, and 5 marks for correct printing of output.
- 10 marks off for using a different structure chart.
- 10 marks off for printing in functions and not in main.

2. This question wants you to trace (that is, execute with hand the way the computer would) through the following simple program (with no functions) and fill in the output of the program in the chart provided below for this question. Show how you arrived at your result for full marks. To show your work, you can show by the side the values of all variables in memory as each instruction is executed. (15 marks)

#include <stdio.h>
int main(void) {

    /* Declare variable */
    int slice=0, knt = 23, flag = 1;
    float cash = 30.50;
    char name[10] = "Kathy", code = 'A';
slice += 203/100 * 12 / 3;
knt = (int)cash % 7;

printf("slice = %d \n", slice);
printf("knt = %d\n", knt);
printf("%s has %c code and %0.2f\n", name, code, cash + 12);

return 0;

}

The output printed by the above program’s 3 printf are:

<table>
<thead>
<tr>
<th>Which printf</th>
<th>Output by Program’s printf is to be written on this column</th>
</tr>
</thead>
<tbody>
<tr>
<td>First printf</td>
<td>slice = 8</td>
</tr>
<tr>
<td>2nd printf</td>
<td>knt = 2</td>
</tr>
<tr>
<td>3rd printf</td>
<td>Kathy has A code and 42.50</td>
</tr>
</tbody>
</table>

Marking Scheme:
Assign –5 marks for each correct tracing of changes and printing of correct values by the printf for a total of 15 marks for the three printf. We allowed a mark of only 2 for each correct tracing with wrong calculation.

3. This question wants you to trace (execute with hand) through a program that has functions and uses call-by-reference, call-by-value as well as some global and local variables. It also wants you to identify important parts of this program as described later.

Given the following program solution to a problem, answer questions 3a to 3c using this solution. Each of the 10 answers in question 3a to 3c is worth 5 marks.

(50 marks total)

#include <stdio.h>

// Global Variables

int num4 = 10, num5=5, num6=2, sum2=0, product2=0;
float average2=0;

/* fn prototypes */
int FindSum (int, int, int);
void FindProd (int, int, int, int *);
void FindAve (int, int, float);

int main (void)
{
    int num1, num2, num3, sum1=0, product1=0;
    float average1=0.0;

    /* Now write the instructions of the control module */

    scanf("%d %d %d", &num1, &num2, &num3);
    // 1st printf in main follows
printf("%d %d %d %d %d %d\n", num1, num2, num3, num4, num5, num6);

sum1=FindSum(num1, num2, num3);
sum2=FindSum(num4, num5, num6);
// 2nd printf in main is next
printf("%d %d\n", sum1, sum2);

FindProd(num1, num2, num3, &product1);
FindProd(num4, num5, num6, &product2);
// 3rd printf in main is next
printf("%d %d\n", product1, product2);

FindAve(sum1, 3, average1);
FindAve(sum2, 3, average2);
// 4th printf in main is next
printf("%.2f %.2f\n", average1, average2);
return 0;

} // end of main

/* Now begin the definitions of the functions */
int FindSum(int first, int second, int third)
{
    int sum;
    sum = (first + second + third) * 2;
    return sum;
}

int divisor = 3; // more global variables

void FindProd (int num1, int num2, int num3, int *prod)
{
    *prod = (num1 * num2 * num3) % divisor;
}

void FindAve (int sum, int knt, float ave)
{
    ave = (float)sum/knt;
}

3a.) With a sample test data 20 5 4 typed at the scanf prompt for the values of the
variables Num1, Num2 and Num3 respectively, write the values printed by the four printf
instructions in main in the table below. Show your work to get full marks here. To show
your work, you can show by the side the values of all variables in memory as each
instruction is executed (that is, show your tracing work).

<table>
<thead>
<tr>
<th>After printf instruction at Position..</th>
<th>Values Printed by CPU in correct order are</th>
</tr>
</thead>
<tbody>
<tr>
<td>First printf in main</td>
<td>20 5 4 10 5 2</td>
</tr>
<tr>
<td>Second printf in main</td>
<td>58 34</td>
</tr>
<tr>
<td>Third printf in main</td>
<td>1 1</td>
</tr>
</tbody>
</table>
3b.) List all global and local variables of the functions (that each function can use) in the table below:

<table>
<thead>
<tr>
<th>Function</th>
<th>Global variables (that can be used by each of the functions)</th>
<th>Local variables (not including its formal parameters)</th>
</tr>
</thead>
<tbody>
<tr>
<td>main</td>
<td>num4, num5, num6, sum2, product2, average2</td>
<td>num1, num2, num3, sum1, product1, average1</td>
</tr>
<tr>
<td>FindSum</td>
<td>num4, num5, num6, sum2, product2, average2=0</td>
<td>sum</td>
</tr>
</tbody>
</table>

3c.) Fill in the correct description of listed parts of the program solution of question 3 in the table below: (Place your answer to each question in the column on its right).

| A function call in the main is .. (provide one complete instruction) | Sum1=FindSum(num1, num2, num3); Sum2=FindSum(num4, num5, num6); FindProd(num1, num2, num3, &product1); FindProd(num4, num5, num6, &product2); FindAve(sum1, 3, average1); FindAve(sum2, 3, average2); |
| A list of formal parameters in the program is. Provide one complete list with the brackets.. | (int first, int second, int third) used in FindSum or (int num1, int num2, int num3, int *prod) used in FindProd, or (int sum, int knt, float ave) used in FindAve. |
| A list of actual parameters in the program is. Provide one complete list with the brackets.. | (num1, num2, num3) or (num4, num5, num6) used in calls to FindSum, or (num1, num2, num3, &product1) or (num4, num5, num6, &product2) used in calls to FindProd, or (sum1, 3, average1) or (sum2, 3, average2) used in calls to finaAve. |
| One function prototype in the program of Problem 3 is .. | int FindSum (int, int, int); or void FindProd (int, int, int, int *); or void FindAve (int, int, float); |

Marking Scheme:
Take 5 marks off for any of the 10 questions in 3a to 3c incorrectly answered. For question 3a, we allow only a maximum mark of 1 (that is – 1.5) for each correct tracing or correct calculation with written answer wrong. For question 3b, we deduct 0.5 mark for each variable in the list missed or for each extra wrong variable included.