60-140
Introduction to Algorithms and Programming I
Dr. Christie Ezeife
**Lab. Exercises #4 Solution (Lab Date: Week 6 of Classes)**

Objectives are to:

1. Keep practising on writing programs with functions as taught in chapter 4 of book. In particular, to feel comfortable with use of call-by-value and call-by-reference parameters.
2. Review all program logic structures as taught in chapter 5 of text book.

Que. 1. Using top-down approach to problem solving with functions and parameters, write a C program to solve the following problem. TrustCard, a company that issues calling cards uses an algorithm to create card numbers. The algorithm reads two four-digit integer numbers and computes a seven digit card number composed such that the first number of the calling card constitutes the four digits from the first input number, and the last three digits are composed from a number obtained by multiplying the sum of the four digits of the first input data by the number 3 and adding the sum of the four digits of the second input data.

Your program should print the following:

1. The calling card number assigned given any two four digit integers.
2. (i.) Original four digit numbers, (ii) their sums, and (iii) the three digit number obtained by multiplying the sum of first number digits by 3, (iv) the final three digit number that makes the last digit of the calling card (that is, (sum of first number digits x 3) + sum of second number digits) .

For example,
Sample Input:

Type the first digit number : 4737
Type the second digit number: 1234

Sample Output:

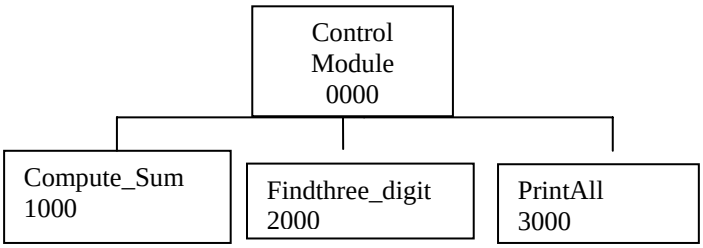
The sum of the first four digits 4737 is : 21
The sum of the 2nd four digits 1234 is : 10
The three digit number 21 x 3 is : 063
The three digit number 063 + 10 is : 073
The calling card number is : 4737073

Thank you for visiting TrustCard company !!

Hints on how to solve

i. Use the following structure chart.

Use the following structure chart for solving the problem.




-All input data should be read in the control module.
-Compute_Sum module should calculate the two sums.
-Findthree_digit should be used to compute the three digits.
-PrintAll should print all output numbers.
Use the integer division and modulus arithmetic to get the digits of the numbers. For example, with Num1 = 4737, and Num2 = 1234, the following equations can be used to get the digits of the number.
Digit4 = Num % 10; ==> Digit4 = 4737 % 10 = 7
Digit3 = (Num / 10) % 10; ==> Digit3 = 4737/10%10 = 473 % 10 = 3
Digit2 = (Num/10) / 10; ==> Digit2 = 4737 / 10 /10 % 10 = 47 % 10 = 7
Digit1 = ((Num/10) / 10)/10; ==> Digit1 = 4737/10/10/10 = 4
Sum1 = Digit1 + Digit2 + Digit3 + Digit4.
Then, after this, the sum of the digits can be obtained the same way as Sum2, and added to the Sum in control module.
A second call to Compute_Sum with the second number can be made to add the sum of the second number to the sum in control module.
The Findthree_digit module can then be called from main to compute the three digit before the Card number is computed in main with the three digit by adding them as the last digits of the first number.
Sum = Sum1 + Sum2.
three_digit = (Sum1 * 3) + Sum2.
Card_num = (Num1 * 1000) + three_digit

Type, compile and run your program and show work in a script file.

**Solution to Que 1**

```
Script started on Thu Sep 02 22:13:01 2010
sol:~/bk2010/programs>cat lab4slnq1.c
#include <stdio.h>

/* Three function prototypes                    */
int Compute_Sum(int);
int Findthree_digit(int, int);
```

```
void PrintAll(int, int, int, int, int, int);
void main( )
{

/* This main function calls three submodules to compute the
 integer calling card number from two four digit integers

 It implements an algorithm for creating calling card numbers for TrustCard
company.  The algorithm reads two four-digit integer numbers and computes a
seven digit card number composed, where the first number of the calling
card constitutes the four digits from the first input number, and the last
three digits are composed from a number obtained by multiplying the sum of
the four digits of the first input data by the number 3 and adding the sum
of the four digits of the second input data.

Program Writer: Dr. Christie Ezeife
Date: Sept. 3, 2010

*/

int Num1, Num2;

int cardnum;
int sum1, sum2, three_digit;
printf("Type the first four digit number               :");
scanf("%d", &Num1);
printf("Type the second four digit number              :");
scanf("%d", &Num2);
sum1 = Compute_Sum(Num1);
sum2 = Compute_Sum(Num2);
three_digit= Findthree_digit(sum1, sum2);
cardnum = (Num1 * 1000) + three_digit;
PrintAll(Num1, Num2, sum1, sum2, three_digit, cardnum);

}  /* end of main   */

int Compute_Sum(int Num)
{
int Digit1, Digit2, Digit3, Digit4, Sum;
Digit1 = Num % 10;
Digit2 = (Num / 10) % 10;
Digit3 = (Num/10) / 10 % 10;
Digit4 =  ((Num/10) / 10)/10;
Sum = Digit1 + Digit2 + Digit3 + Digit4;

return (Sum);
```

```
}

int Findthree_digit(int Sum1, int Sum2)
{
  int three_digit;
/* Now compute the three digit  */
three_digit = (Sum1 * 3) + Sum2;
return (three_digit);

} /*  Find_threedigit ends*/
/* Definition of PrintAll */
void PrintAll(int Num1, int Num2, int Sum1, int Sum2, int three_digit, int Cardnum)
{
      int first3digit;
       first3digit = three_digit - Sum2;

      printf("The sum of the first four digits  %d is \t: %d \n", Num1, Sum1);
      printf("The sum of the 2nd four digits %d is    \t: %d \n", Num2, Sum2);
      printf("The three digit number %d  x  3 is \t\t: %03d \n", Sum1,  first3digit);
      printf("The three digit number %03d  +  %d is \t\t: %03d \n", first3digit, Sum2, three_digit);

      printf("The calling card number is  \t\t\t: %d   \n", Cardnum);
      printf("\nThank you for visiting TrustCard company !! \n");

}

sol:~/bk2010/programs>cc lab4slnq1.c
sol:~/bk2010/programs>a.out
Type the first four digit number              :4737
Type the second four digit number             :1234
The sum of the first four digits  4737 is     : 21
The sum of the 2nd four digits 1234 is        : 10
The three digit number 21  x  3 is            : 063
The three digit number 063  +  10 is          : 073
The calling card number is                    : 4737073

Thank you for visiting TrustCard company !!
sol:~/bk2010/programs>exit
exit
script done on Thu Sep 02 22:13:56 2010
```

**Solution to Que 2**

Two types of parameter passing techniques are call-by-reference and call-by value parameters. With call-by-reference parameters, the address of the memory location of the variable in the calling module is passed to the called module, thereby giving it the permission to modify the content of this address as it pleases. However, with call-by-value parameter, only a copy of the content of this memory variable is passed from the calling module to a called module, making for data protection of the content of this memory address in the called module.

Que 3. Why are cohesion and coupling important to programmers?

**Solution to Que 3**:

Cohesion allows a big problem to be split into small independent modules which different programmers can work on in parallel thereby producing quicker results. Coupling allows the results of the independent modules to be integrated well together towards solving the main program.

Que 4. Name all the program logic structures, giving a simple example of each structure.

**Solution to Que 4**:

There are three programming logic structures namely:

a. Sequential logic structure: This is the default logic structure which requires that instructions in a program or algorithm be executed from top to bottom in sequence with instruction 1 executed first, followed by execution of instruction 2, then instruction 3 and so on. Some algorithmic and program instructions are of sequential type and these are assignment instructions, read (scanf) and print (printf) instructions and function calls. I give example of each of these types of instructions.

Assignment instruction:

```
Sum = Sum + Num;
Sum += Num;
```

Read instruction

```
Read (Num);          ----(in an algorithm)
scanf("%f", &Num);   ---(in a C program)
```

Print instruction

```
Print (Sum);           -----(in an algorithm)
printf("%f", Sum);     ----(in a C program)
```

Function call:

```
CalcSum (Num, Sum);     ----(in an algorithm)
CalcSum(Num, &Sum);     ----(in a C program)
```

b. Decision logic structure: This logic structure requires that one among a number of alternative execution path be followed depending on the result of a test condition. This logic structure is implemented in programs and algorithms using “if” and “switch-case” instructions.

if instruction as used in both an algorithm and a C program:

```
if  (Num1 > Num2)
    Larger = Num1;
  else
    Larger = Num2;
```

switch-case Instruction as used in a C progam:

```
swith (grade ) {
   case  ‘A’ : printf (" Excellent student");
   case ‘B’  : printf (" Very good student");
   case ‘C’  : printf (" Good student " );
   case ‘D’  : printf ("Ok student ");
default:      printf ("poor or unknown");
}
```

b. Repetition Logic Structure: The instructions in this logic structure require that a number of instructions be executed continuously in a loop until a certain termination condition is satisfied. The program and algorithmic instructions that are of repetitive type are while instruction, do-while instruction and for loop instruction.

while instruction:

```
   Sum = 0;
   knt = 0;
   scanf (“%d”, &Num);
   while (knt < 10)
      {
          Sum += Num;
          Knt++;
          scanf (“%d”, &Num);
      }
*****
```

do-while instruction:

```
   Sum = 0;
   Knt = 0;
   scanf (“%d”, &Num);
   do
          {
```

```
        Sum += Num;
        Knt++;
        scanf ("%d", &Num);

        } while (knt < 10)
```

*******

for- loop instruction

```
for (knt = 0; knt<10; knt++)
  {
      scanf ("%d", &Num);
        Sum += Num;
}
```

Que. 5. Name the major types of modules in a solution and explain their functions.

**Solution to Que 5**:

Some types of modules we may create in our solution when using top-down design approach are:

a) Control Module or Main Driver: which shows the overall flow of the problem and uses or calls other modules.
b) Init Module: may be needed if a large set of variables like arrays elements and others need to be introduced e.g.; sum=0, knt=1
c) Read Data Module: may be needed if a large set of variables like array elements needs to be read into memory. Data validation may need to be done in this module as well.
d) Calculation Modules: for arithmetic calculation, string manipulations like sorting, etc.
a) File Maintenance Modules: could be used for adding, changing or deleting a record in file.
f) Print Result Module for printing ouput of program.
g) Wrap-up Module: may be used for printing totals after loop instructions, and for closing files.