Assignment 3

Zhang Junming 1301058

**Exercise 1**

Software Development Process

1. Problem statement:

Write a program to calculate the sum of the odd digits and even digits of an entered integer.

1. Analysis:

Inputs: real number to represent the value of digital.

Outputs: sum of the entered number’s odd digits; sum of the entered number’s even digits.

Additional requirements or constraints: “int” cannot exceed 10 characters.

1. Design

Algorithm:

1. int **a** -represents the entered number(for judge the order of units).

int **b-** represents the sum of even numbers.

int **c-** represents the sum of odd numbers.

int **d-** represents the entered number(for get value of each digital).

int **e-** represents the remainder.

int **i&j-** represents the numerical digit.

2. Ask user input a real number and remind that this real number cannot exceed 10 characters, otherwise this program will automatically end.

3. Read and store the value of entered number **a**

4. Judge whether the value of **a** within 10 digitals.

If is, a=d; if not, ending this program.

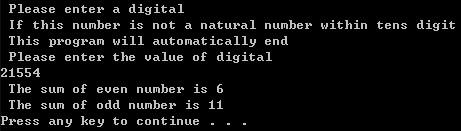
5. Setting up a circulation using “for” and “if else” functions for judge the numerical digit.

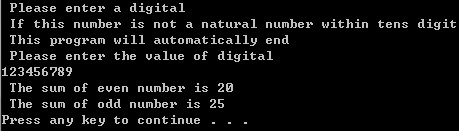
6. Setting up another circulation using “for” and “if else” functions. Let program judge and calculate the sum of even and odd digitals.

7. Display the sum of even and odd number.

8. Setting up a pause process using “system pause” function.

1. Implementation: see the C code in file exercise1.c with comments.
2. Testing: the C program was tested by carrying out a set of experiments and the C program output was verified successfully. For instance,





**Exercise 2**

Software Development Process

1. Problem statement:

Write a C program that prints out all different possibilities of obtaining 2 pounds using coins of values 2 pence, 5 pence, and 10 pence. Indicate also how many possibilities have been found.

1. Analysis:

Input: program

Outputs: all different possibilities of obtaining 2 pounds using coins of values 2 pence, 5 pence, and 10 pence; how many possibilities have been found.

Additional requirements or constraints: number of the possibilities is calculated by C program.

1. Design:

Algorithm:

1. int **a,b,c** - represent the amounts of 2 pence,5 pence and 10 pence.

Int **i**- represents the possible answer.

2. Setting up 3 circulations of the amounts of 2,5 and 10 pence to calculate all possible situations.

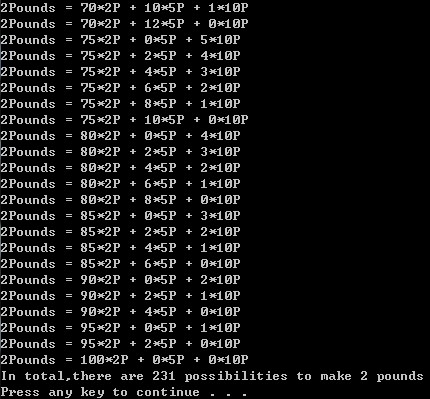
3. Judge whether all situations conform to the formula.

4. Display all required situations.

5. Using i++ to statistics all required situations.

6. Setting up a pause process using “system pause” function.

1. Implementation: see the C code in file exercise2.c with comments.
2. Testing: the C program was tested by carrying out a set of experiments and the C program output was verified successfully. For instance,



**Exercise 3**

Software Development Process

1.Problem statement:

Write a C program evaluates the square and square root of a series of numbers and displays the results. If the entered number is negative, just display the square. If the entered number is 7777, stop this program and display how many items have input and how many of them are negative.

2.Analysis:

Input: some positive or negative number, 7777 to stop this program.

Outputs: the square and square roof of each positive number and the square of each negative number, when input 7777, stop program and display how many items have input and how many of them are negative.

Additional requirements or constraints: 7777 is the end command. When entered number is negative, the square root should display “Number is negative”.

3.Design:

Algorithm:

1. float **a**– represents the value of entered number.

float **b,c**– represent the value of square and square root.

int **i,j**– represent the number of positive and negative items.

int **k**– represent the sum of items

1. Setting up a ‘do-while’ circulation and stop when a=7777 .

3. Ask user to enter a number and enter 7777 to stop the program.

4. Read and store the value of entered number **a**

5. Calculate square of **a** and square root of **a**

6. Judge when **a** is not 7777, display the square of **a**

7. Judge when **a** bigger than 0 and not equal 7777, display the square root of **a**

8. Judge when **a** smaller than 0 and not equal 7777, display “number is negative.

9. Setting up a “if” function, when **a** bigger than 0, i equal i++, when **a** smaller than 0,j equal j++.

10. Out of loop, calculate the sum of positive and negative items.

11.display how many items done and how many of them are negative.

11. Setting up a pause process using “system pause” function

4.Implementation: see the C code in file exercise3.c with comments.

1. Testing: the C program was tested by carrying out a set of experiments and the C program output was verified successfully. For instance,

Show in next page

