北 京 林 业 大 学

2020 学年— 2021 学年第 一 学期软件工程实验报告书

专 业：计算机科学与技术(创新实验班)　　班　　级：18

姓　　名：连月菡　　学　　号：181002222

实验地点：计算中心　　任课教师：赵方

实验题目：实验四、软件设计

实验环境：一台装有 MS VISIO 和 MS WORD 软件的 PC 机

实验内容：

（1） 了解软件测试基本概念和基本过程。
（2） 掌握使用等价类划分法和边界值分析法进行测试用例设计的基本步骤。

实验目的：

假设商店货品价格（R）都不大于 100 元，且货品价格均为整数，若顾客付款（P）在 100 元以内（含 100 元，且为整数）。现有一个程序能在每位顾客付款后给出找零钱的最佳组合（找给客户货币张数最少）。假定此商店的货币面值只包括：50 元（N50）、10 元（N10）、5 元（N5）、1 元（N1）四种。
请结合等价类划分法和边界值分析法为上述程序设计出相应的测试用例。试用 C 语言编写具有上述功能的程序；执行程序并使用设计好的测试用例对程序进行测试并收集测试结果。

实验指导:

（1）输入条件 R 的有效等价类、无效等价类、边界值
a)商品价格 R 为整数，有效等价类： 0<R<=100 ； 无效等价类：R<=0,R>100
R 的边界值： 0；100，101
b)商品价格 R 为实数（无效等价类）
c)商品价格 R 为字符（无效等价类）
（2）输入条件 P 的有效等价类、无效等价类、边界值
a)顾客付款 P，有效等价类： 由 0<P<=100, R<=P 得 R<=P<=100；无效等价类：P<R,P>100
P 的边界值：-1,0；100，101
b) 顾客付款 P 为实数（无效等价类）
c) 顾客付款 P 为字符（无效等价类）
（3）输出条件的等价类（N50、N10、N5、N1）

考虑输出零钱的条件范围和票面个数；

（4）利用等价类划分测试用例设计表，根据输入数据 R、P 等价类和边界值的组合设计测试用例覆盖所有输入和输出等价类。

实验要求：

（1） 按照测试策略要求分别使用边界值分析、等价类划分方法设计相应测试用例。

（2） 严格按照设计步骤设计测试用例，设计过程应该使用等价类划分表格辅助设计。

（3） 注意等价类划分的完整性。

实验结果：

题目分析：R 为应收金额，P 为实收金额

输入情况:R>100，0<R<=100，0<R，P>100，R<=P<=100，P<R。

输出情况:面值 50 元张数=1 或 0，0<= 面值 10 元张数 <5，面值 5 元张数=1||0，0<= 面值 1 元张数<5。

有效等价类，0<R<=P<=100

P 和 R 的关系	找零钱情况
P-R>=50	至少有 1 张 50 元
40<=[(P-R)-其他大于 10 元面额的纸币总额]<50	有 4 张 10 元
5<=[(P-R)-其他大于 5 元面额的纸币总额]<10	有 1 张 5 元
[(P-R)-其他大于 1 元面额的纸币总额]==4	有 4 张 1 元

无效等价类

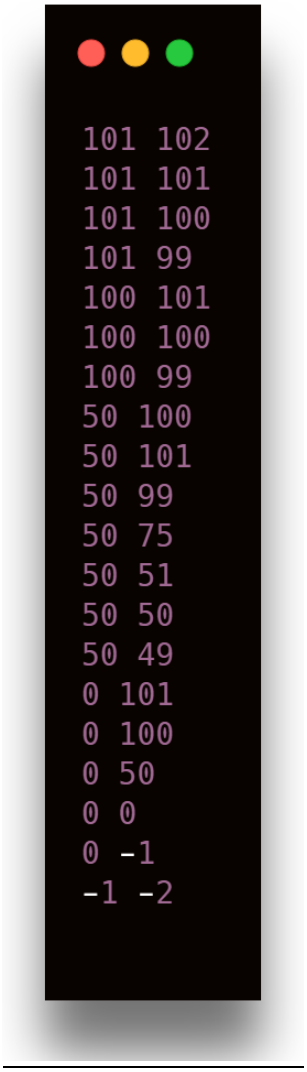
P	R
P<0	R 为任意值
0<P<R	R>0
P>100	R 为任意值
P 为任意值	R<0
P 为任意值	R>100

测试实例（R，P）有：

用例编号	R	P	预期输出结果
1	101	102	非法输入
2	101	101	非法输入
3	101	100	非法输入
4	101	99	非法输入

5	100	101	无需找钱
6	100	100	无需找钱
7	100	99	非法输入
8	50	100	非法输入
9	50	101	50~1,10~0,5~0,1~0
10	50	99	50~0,10~4,5~1,1~4
11	50	75	50~0,10~2,5~1,1~0
12	50	51	50~0,10~0,5~0,1~1
13	50	50	无需找钱
14	50	49	非法输入
15	0	101	非法输入
16	0	100	非法输入
17	0	50	非法输入
18	0	0	非法输入
19	0	-1	非法输入
20	-1	-2	非法输入

测试用例:


101 102
101 101
101 100
101 99
100 101
100 100
100 99
50 100
50 101
50 99
50 75
50 51
50 50
50 49
0 101
0 100
0 50
0 0
0 -1
-1 -2

测试代码：

主函数

```
int main() {
    /*
    * 1. 定义变量
    */
    int money[] = { 50,10,5,1 };//存放可供找零的面值，降序排列
    int k = sizeof(money) / sizeof(money[0]);//可供找零的面值种类数

    /*
    * 2. 在while循环中，每次循环进行一次找零钱的操作
    */
    while (1) {
        int real = 0, input = 0;//应付商品总金额，顾客付的金额
        char s[101];
        int gap = 0, end = 0;//检测两个数字间隔的位置
        printf("输入应付商品总金额和顾客实际支付金额，用空格隔开：\n");
        gets_s(s);//读取字符串

        for (int i = 0; i < 101; ++i) {//遍历
            if (s[i] == ' ')
                gap = i;
            if (s[i] == '\0') {
                end = i;
                break;
            }
        }
        real = change(0, gap, s);//应付商品总金额
        input = change(gap + 1, end,  s);//顾客实际支付金额

        printf("%d %d", real, input);//输入应付商品总金额，顾客付的金额
        if (real > 100 || input > 100 || real <= 0 || input <= 0) {
            printf("非法输入!\n");
            continue;
        }
        GreedMoney(money, k, input - real);
    }

}
```

对于所谓的标准硬币系统，如美国和许多其他国家所使用的硬币系统，贪婪算法会选择最大面额的硬币，该硬币的面额不超过要找零钱的剩余数量，将产生最佳结果。但是，对于任意硬币系统而言并非如此。 例如，如果硬币面额为 1、3 和 4，则要得到 6，贪心算法将选择三个硬币（4,1,1），而最优解为两个硬币（3,3），但对于本程序而言， 1， 5， 10， 50 刚好符合美国货币系统的硬币面值， 因此使

用贪心算法不会出现上述谬误。

其他函数

```
/*3.利用贪心算法，传入 货币面值数组，可供找零的面值种类数，
*  顾客实际支付金额与应付商品总金额之差
*/
void GreedMoney(int m[], int k, int n) {
    if (n == 0) {
        printf("无需找钱!\n"); return;
    }
    printf("%s", n > 0 ? "应该找" : "顾客付钱金额不足!");

    int num[] = { 0,0,0,0 };
    int money[] = { 50,10,5,1 };//存放可供找零的面值，降序排列
    for (int i = 0; i < k; i++)
        while (n >= m[i]) {//当应找金额大于当前最大面额货币
            switch (m[i]) {
            case 50:
                num[0]++;  break;
            case 10:
                num[1]++;  break;
            case 5:
                num[2]++;  break;
            case 1:
                num[3]++;  break;
            }
            n = n - m[i];//继续寻找顾客剩余的钱
        }
    for (int j = 0; j < 4; ++j)//输出提示
        if (num[j] > 0)
            printf("%d张%d元 ", num[j], money[j]);
    printf("\n");
}
/*4. 传入字符数组的数字字符的起始位置，
*  结束位置，输入的字符串数组
*/
int  change(int beg, int pos,  char s[]) {//字符转变为数字
    int j = beg;
    int num = 0;
    while (j < pos) {
        num = num * 10 + (s[j++] - '0');
    }
    return num;
}
```

测试用例输出结果：

```
输入应付商品总金额和顾客实际支付金额，用空格隔开：
101 102
非法输入！
输入应付商品总金额和顾客实际支付金额，用空格隔开：
101 101
非法输入！
输入应付商品总金额和顾客实际支付金额，用空格隔开：
101 100
非法输入！
输入应付商品总金额和顾客实际支付金额，用空格隔开：
101 99
非法输入！
输入应付商品总金额和顾客实际支付金额，用空格隔开：
100 101
非法输入！
输入应付商品总金额和顾客实际支付金额，用空格隔开：
100 100
无需找钱！
输入应付商品总金额和顾客实际支付金额，用空格隔开：
100 99
顾客付钱金额不足！
输入应付商品总金额和顾客实际支付金额，用空格隔开：
50 100
应该找1张50元
输入应付商品总金额和顾客实际支付金额，用空格隔开：
50 101
非法输入！
输入应付商品总金额和顾客实际支付金额，用空格隔开：
50 99
应该找4张10元 1张5元 4张1元
输入应付商品总金额和顾客实际支付金额，用空格隔开：
50 75
应该找2张10元 1张5元
输入应付商品总金额和顾客实际支付金额，用空格隔开：
50 51
应该找1张1元
输入应付商品总金额和顾客实际支付金额，用空格隔开：
50 50
无需找钱！
```

```
输入应付商品总金额和顾客实际支付金额，用空格隔开：
50 49
顾客付钱金额不足！
输入应付商品总金额和顾客实际支付金额，用空格隔开：
0 101
非法输入！
输入应付商品总金额和顾客实际支付金额，用空格隔开：
0 100
非法输入！
输入应付商品总金额和顾客实际支付金额，用空格隔开：
0 50
非法输入！
输入应付商品总金额和顾客实际支付金额，用空格隔开：
0 0
非法输入！
输入应付商品总金额和顾客实际支付金额，用空格隔开：
0 -1
非法输入！
输入应付商品总金额和顾客实际支付金额，用空格隔开：
-1 -2
非法输入！
```