1) Finding the bug

I wrote a random pattern generator C++ program to generate random input patterns. This program generator is present in debug/rand.cpp

Get Started | rand.cpp

rand.cpp > ...

```
3   #include<stdlib.h>
4   #include<time.h>
5   #include <cstdlib>
6
7   #define min 1
8   #define max 3
9
10  using namespace std;
11
12  void genPat(ofstream &testPat)
13  {
14      for(int k=0;k<5;k++)
15      {   int randNum = rand()%(max-min + 1) + min;
16          if(randNum==1)
17          {
18          testPat <<"01";
19          }
20          if(randNum==2)
21          {
22          testPat <<"10";
23          }
24          if(randNum==3)
25          {
26          testPat <<"11";
27          }
28      }
29      testPat<<"1"<<endl;
30  }
31
32  int main(int argc,char* argv[])
33  {   ofstream testPat;
34      testPat.open("Test.pattern");
35      srand(time(NULL));
36      int max_iter=atoi(argv[1]);
37      testPat <<max_iter<<" 11"<<endl;
38      testPat <<"XXXXXXXXXX0"<<endl;
39      int count=1;
```

I generated some random input patterns and observed that for a case, p was triggered to be 1. I started debugging & found out that there exists a bug for the case when the machine didn't have enough change coins and it should return all the money ideally, however it didn't return the money and rather ate all the money.

To demonstrate this bug case, I give the machine input of three 10 dollars and I want to buy product C which costs 22 dollars. Since, input_money > service_value, the transaction should take place and the vending machine should return 8 dollars as change (One 5 dollar and three 1 dollars). But it's specified that the machine has initially just 2 coins initially.

```
countNTD_50      <= 3'd2;
countNTD_10      <= 3'd2;
countNTD_5       <= 3'd2;
countNTD_1       <= 3'd2;
```

So, I passed the input pattern as specified above
Case I) Input 30 dollars (Case present in bug1.pattern, bug1.output)

Input_pattern

```
 1 38 11
 2 XXXXXXXXXX0
 3 11000011001
 4 00XXXXXXXX1
 5 00XXXXXXXX1
 6 00XXXXXXXX1
 7 00XXXXXXXX1
 8 00XXXXXXXX1
 9 00XXXXXXXX1
10 00XXXXXXXX1
11 00XXXXXXXX1
12 00XXXXXXXX1
```

Bug observed in output

```
 1 00000000000000000
 2 01000000000000000
 3 10110000000000000
 4 10110000000000000
 5 10110000000000000
 6 10110000000000000
 7 10110000010000000
 8 10110000010000000
 9 10110010010000000
10 10110100010000000
11 00000000000000001
```

We should get back our input 30 dollars since the machine couldn't provide change, however we can see that we got back 0 dollars.

Case II) Input 25 dollars (Case present in bug.pattern, bug.output)

To re-verify I gave input 25 dollars, I observed the same bug as it is the same case of not able to exchange initially.

```
 1 38 11
 2 XXXXXXXXXX0
 3 11000110001
 4 00XXXXXXXX1
 5 00XXXXXXXX1
 6 00XXXXXXXX1
 7 00XXXXXXXX1
 8 00XXXXXXXX1
 9 00XXXXXXXX1
10 00XXXXXXXX1
11 00XXXXXXXX1
```

```
 1 00000000000000000
 2 01000000000000000
 3 10110000000000000
 4 10110000000000000
 5 10110000000000000
 6 10110000000000000
 7 10110000000000000
 8 10110010000000000
 9 10110100000000000
10 00000000000000001
```

Now, that a bug is found, the next question arises is how to fix it

2) Fixing the Bug

This bug seems to be the case of state being as 'SERVICE_OFF" during the exchange stage. By changing "**serviceTypeOut_w = `SERVICE_BUSY**", we can force the machine to go in the state SERVICE_BUSY thus empowering it to first exchange the input and then jump to the output state. I have taken the waiting cycle long enough to carry out the exchange.

The code should be modified as per line 236

```
222             `NTD_1  : begin
223               if (serviceValue >= `VALUE_NTD_1) begin
224                 if (countNTD_1 == 3'd0) begin
225                   serviceValue_w    = inputValue;
226                   itemTypeOut_w     = `ITEM_NONE;
227                   serviceCoinType_w = `NTD_50;
228                   countNTD_50_w     = countNTD_50 + coinOutNTD_50;
229                   countNTD_10_w     = countNTD_10 + coinOutNTD_10;
230                   countNTD_5_w      = countNTD_5 + coinOutNTD_5;
231                   countNTD_1_w      = countNTD_1 + coinOutNTD_1;
232                   coinOutNTD_50_w   = 3'd0;
233                   coinOutNTD_10_w   = 3'd0;
234                   coinOutNTD_5_w    = 3'd0;
235                   coinOutNTD_1_w    = 3'd0;
236                   serviceTypeOut_w  = `SERVICE_BUSY;
```

After simulating the patterns with the fixed code, I verified that the machine returns the correct change.

```
 1 0000000000000000
 2 0100000000000000
 3 1011000000000000
 4 1011000000000000
 5 1011000000000000
 6 1011000000000000
 7 1011000000000000
 8 1011001000000000
 9 1011010000000000
10 1000000000000000
11 1000000000000000
12 1000000000010000
13 1000000000100000
14 1000000000100000
15 1000000010100000
16 1000000010100000
17 0000000010100000
```

Case: Input 25 dollars

```
 1 0000000000000000
 2 0100000000000000
 3 1011000000000000
 4 1011000000000000
 5 1011000000000000
 6 1011000000000000
 7 1011000010000000
 8 1011000010000000
 9 1011001001000000
10 1011010001000000
11 1000000000000000
12 1000000000000000
13 1000000000010000
14 1000000000100000
15 1000000000110000
16 1000000000110000
17 1000000000110000
```

Case: Input 30 dollars

We can see correct change is exchanged in both the cases; thus I fixed the bug.