

Capital University of Science and Technology

Department of Software Engineering

Name:Hafiz Naqash,Mohammad Khayyam

Reg-Num:Bse181057,Bse181055

Section:01

Assignment:03

Submitted-To:Sir.Sameer Obaid

**Table of content:**

Case Study:

1.introduction.

2.Discription.

Functions:

1. Void deposit(double ammount,int range,int num\_deposit);

2. Void login(int pin);

3.Void withdraw(double amount);

Equalence class partitioning:

1.Strong Robust class partitioning

2.Test Cases

2.1 test Cases Of void withdraw(double amount)

2.2 Void login(int pin)

2.3 Void deposit(double ammount,int range,int num\_deposit

Cause-Effect Graph:

1.Cause and Effect

2. Graphs

3.Decision Table

4.Test Cases;

4.1 Choosing BVA

4.2 Identifying Testcases

4.3 draw a table for Identifying Test Cases

Summary Of Changes:

1: Changes made on basis of feedback mentioned in assignment 01

1.1: We are change the invalid fuction void Transection(double Amount) as you say in

Assignment 1 into Function void login(int pin).

**Case Study Automated Machine ATM**

**Introduction:**

The Automated machine (ATM) is an automatic banking machine (ABM) that allows the customer to complete basic transactions without any help from bank representatives. The basic one allows the customer to only draw cash and receive a report of the account balance.

**Discription:**

The software to be designed will control a simulated automated teller machine(ATM) having a magnetic stripe reader for reading an ATM card, a customer An account is accessible through a cash card.console (keyboard and display) for interaction with the customer, a slot for depositing envelopes, a dispenser for cash (in multiples of $20), a printer fo printing customer receipts, and a key-operated switch to allow an operator to start or stop the machine. The ATM will communicate with the bank’s computer over an appropriate communication link. The ATM will service one customer at a time. A customer will be required toinsert an ATM card and enter a personal identification number (PIN) - both ofwhich will be sent to the bank for validation as part of each transaction. Thecustomer will then be able to perform one or more transactions. The card willbe retained in the machine until the customer indicates that he/she desires nofurther transactions, at which point it will be returned.The ATM must be able to provide the following services to the customer: I A customer must be able to make a cash withdrawal from any suitable account linked to the card, in multiples of $20.00. Approval must be obtained from the bank before cash is dispensed.I A customer must be able to make a deposit to any account linked to the card, consisting of cash and/or checks in an envelope. The customer will enter the amount of the deposit into the ATM, subject to manualverification when the envelope is removed from the machine by an operator. Approval must be obtained from the bank before physically

accepting the envelope. I A customer must be able to make a transfer of money between any two accounts linked to the card. I A customer must be able to make a balance inquiry of any account linked to the card.A customer must be able to abort a transaction in progress by pressing the Cancel key instead of responding to a request from the machine. The ATM will communicate each transaction to the bank and obtain verification that it was allowed by the bank. Ordinarily, a transaction will be considered complete by the bank once it has been approved. In the case of a deposit, a second message will be sent to the bank indicating that the customer has deposited the envelope. (If the customer fails to deposit the envelope within the timeout period, or presses cancel instead, no second message will be sent to the bank and the deposit will not be credited to the customer.

**Functions:**

1.Void deposit(double ammount,int range,int num\_deposit);

2.Void login(int pin);

3.Void withdraw(double amount);

**Equilence Class patitioning:**

Equilence Class partitioning (ECP) is a software testing technique that divides the input data of a software unit into partitions of equivalent data from which test cases can be derived.

**Strong robust equivalence class partitioning:**

. In which we are testing all combination of inside the boundary as well as outside the boundary of input value.

**Test Cases:**

## Function 1: Void Withdraw (double amount):

Input Values=1>=amount<=10

So the min<0 , min+1=1 , normalValue=5 , max=10 , max>11

|  |  |  |
| --- | --- | --- |
| **TestCase** | **Amount** | **Output** |
| 1 | 0 | Invalid |
| 2 | 1 | Valid |
| 3 | 5 | Valid |
| 4 | 9 | Valid |
| 5 | 11 | InValid |

## Function 2: Void login(int pin):

Input values: 1>=pin<=10

So the min<0, min+1=2, normal Value=5, max=10, max>11

|  |  |  |
| --- | --- | --- |
| **TestCase** | **Amount** | **Outputs** |
| 1 | 0 | Invalid |
| 2 | 1 | Valid |
| 3 | 5 | Valid |
| 4 | 10 | Valid |
| 5 | 11 | InValid |

## Function 3: Void Deposit(double ammount,int range,int num\_deposit);

Input Values=**1>=amount<=30;**

**1>=rang<=30;**

**1>=num\_deposit<=30;**

Input Values: min<0 , min +1=1 , Normal Value=15, max=30 , max>31

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **TestCase** | **amount** | **rang** | **Num\_deposit** | **Output** |
| 1 | 0 | 0 | 0 | invalid |
| 2 | 0 | 0 | 1 | invalid |
| 3 | 0 | 0 | 2 | invalid |
| 4 | 0 | 0 | 3 | invalid |
| 5 | 0 | 0 | 4 | invalid |
| 6 | 0 | 0 | 5 | invalid |
| 7 | 0 | 0 | 6 | invalid |
| 8 | 0 | 0 | 7 | invalid |
| 9 | 0 | 0 | 8 | invalid |
| 10 | 0 | 0 | 9 | invalid |
| 11 | 0 | 0 | 10 | invalid |
| 12 | 0 | 1 | 1 | valid |
| 13 | 0 | 1 | 2 | valid |
| 14 | 0 | 1 | 3 | valid |
| 15 | 0 | 1 | 4 | valid |
| 16 | 0 | 1 | 5 | valid |
| 17 | 0 | 1 | 6 | valid |
| 18 | 0 | 1 | 7 | valid |
| 19 | 0 | 1 | 8 | valid |
| 20 | 0 | 1 | 9 | valid |
| 21 | 1 | 1 | 0 | valid |
| 22 | 1 | 1 | 1 | valid |
| 23 | 1 | 2 | 2 | invalid |
| 24 | 1 | 3 | 3 | valid |
| 25 | 1 | 4 | 4 | invalid |
| 26 | 1 | 5 | 5 | valid |
| 27 | 1 | 6 | 6 | valid |
| 28 | 1 | 7 | 7 | valid |
| 29 | 1 | 8 | 8 | valid |
| 30 | 1 | 9 | 9 | valid |
| 31 | 1 | 10 | 10 | valid |
| 32 | 1 | 11 | 11 | valid |
| 33 | 1 | 12 | 12 | valid |
| 34 | 1 | 13 | 13 | valid |
| 35 | 1 | 14 | 14 | valid |
| 36 | 1 | 15 | 15 | valid |
| 37 | 1 | 16 | 16 | valid |
| 38 | 1 | 17 | 17 | valid |
| 39 | 1 | 18 | 18 | invalid |
| 40 | 15 | 0 | 0 | valid |
| 41 | 15 | 1 | 0 | valid |
| 42 | 15 | 2 | 2 | invalid |
| 43 | 15 | 3 | 2 | valid |
| 44 |  | 4 | 3 | valid |
| 45 | 15 | 5 | 4 | valid |
| 46 | 15 | 6 | 5 | valid |
| 47 | 15 | 7 | 6 | valid |
| 48 | 15 | 8 | 7 | valid |
| 49 | 15 | 9 | 8 | valid |
| 50 | 15 | 10 | 9 | invalid |
| 51 | 15 | 11 | 10 | invalid |
| 52 | 15 | 12 | 12 | valid |
| 53 | 15 | 13 | 12 | invalid |
| 54 | 15 | 13 | 13 | valid |
| 55 | 15 | 14 | 14 | valid |
| 56 | 15 | 15 | 15 | valid |
| 57 | 15 | 16 | 16 | valid |
| 58 | 15 | 17 | 17 | valid |
| 59 | 15 | 18 | 18 | valid |
| 60 | 15 | 19 | 19 | valid |
| 61 | 15 | 20 | 20 | valid |
| 62 | 15 | 21 | 21 | valid |
| 63 | 15 | 22 | 22 | valid |
| 64 | 15 | 23 | 23 | valid |
| 65 | 31 | 0 | 0 | valid |
| 66 | 31 | 31 | 36 | valid |
| 67 | 31 | 30 | 31 | valid |
| 68 | 31 | 0 | 0 | invalid |
| 69 | 31 | 1 | 1 | invalid |
| 67 | 31 | 2 | 2 | invalid |
| 68 | 31 | 3 | 3 | valid |
| 69 | 31 | 4 | 4 | valid |
| 70 | 31 | 6 | 6 | valid |
| 71 | 31 | 7 | 7 | valid |
| 72 | 31 | 8 | 8 | valid |
| 73 | 31 | 9 | 9 | valid |
| 74 | 31 | 10 | 10 | valid |
| 75 | 31 | 11 | 11 | valid |
| 76 | 31 | 12 | 12 | invalid |
| 77 | 31 | 13 | 13 | invalid |
| 78 | 31 | 14 | 14 | invalid |
| 79 | 31 | 31 | 31 | valid |
| 80 | 31 | 30 | 35 | valid |
| 81 | 31 | 20 | 20 | valid |
| 82 | 31 | 19 | 19 | valid |
| 83 | 31 | 25 | 35 | valid |
| 84 | 31 | 26 | 26 | valid |
| 85 | 31 | 27 | 27 | valid |
| 86 | 31 | 28 | 28 | valid |
| 87 | 31 | 29 | 29 | valid |

Cause Effect Graph:

Causes:

C1: Amount=>500

C2: Multiple of 500

C3:Amount <50000

C4: Amount < user Account Balance

C5:valid card

C6:valid pin

Effects:

E1: invalid amount entered

E2: Successful transection

E3:Successful login

Graphs:

Graph 1:

^

Graph 2:

^

^

^

Decision Table:

|  |  |  |
| --- | --- | --- |
| Causes | 1 | 2 |
| C1 | 0 | 1 |
| C2 | 0 | 1 |
| C3 | 0 | 1 |
| C4 | 0 | 1 |
| C5 | 1 | 0 |
| C6 | 1 | 0 |
| Effects |  |  |
| E1 | 0 | 1 |
| E2 | 0 | 1 |
| E3 | 1 | 0 |

Test Cases:

We are choosing BVA:

Identifying Test Cases:

1.Void deposit(int amount)

Min= 500

Min+1= 1000

Normal= 25,000

Max= 50,000

Max-1= 49,000

Test Cases Table

|  |  |  |
| --- | --- | --- |
| Test Cases | Test data | Expected Outputs |
| 1 | 5000 | valid |
| 2 | 1000 | valid |
| 3 | 25000 | valid |
| 4 | 50000 | valid |
| 5 | 49000 | valid |

2.Void login(int pin)

Min= 1

Min+1= 2

Normal= 5

Max= 10

Max-1= 9

Test Cases Table

|  |  |  |
| --- | --- | --- |
| Test Cases | Test data | Expected Outputs |
| 1 | 1 | valid |
| 2 | 11 | valid |
| 3 | 45615 | valid |
| 4 | 1236789322 | valid |
| 5 | 21389321 | valid |