Capital University of Science and Technology

Department of Software Engineering

Name:Hafiz Naqash,Mohammad Khayyam

Reg-Num:Bse181057,Bse181055

Section:01

Assignment:02

Submitted-To:Sir.Sameer Obaid

Table of content:

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 transection(double amount);

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:

1. **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

2. **Function 2: Void Transection (**double amount**):**

Input values: 40>=amount<=80

So the min<39, min+1=40, normal Value=60, max=80, max+1>81

TestCase	**amount**	**Outputs**
1	39	Invalid
2	40	Valid
3	60	Valid
4	79	Valid
5	81	InValid

3. **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+1>31

TestCase	**amount**	**rang**	**Num_deposit**	**Output**
1	0	0	0	Equalent
2	0	0	1	Not a triangle
3	0	0	2	Not a triangle
4	0	0	3	Not a triangle
5	0	0	4	Not a triangle
6	0	0	5	Not a triangle
7	0	0	6	Not a triangle
8	0	0	7	Not a triangle
9	0	0	8	Not a triangle
10	0	0	9	Not a triangle
11	0	0	10	Not a triangle
12	0	1	1	isoscles
13	0	1	2	Scaline
14	0	1	3	Scaline
15	0	1	4	Scaline
16	0	1	5	Scaline
17	0	1	6	Scalinr
18	0	1	7	Scaline
19	0	1	8	IScaline
20	0	1	9	Scaline
21	1	1	0	Isoscles
22	1	1	1	Eqvalent

23	1	2	2	Scaline
24	1	3	3	Isoscles
25	1	4	4	scaline
26	1	5	5	isoscles
27	1	6	6	isosceles
28	1	7	7	scaline
29	1	8	8	isoscles
30	1	9	9	scaline
31	1	10	10	scaline
32	1	11	11	isosles
33	1	12	12	isoscles
34	1	13	13	scaline
35	1	14	14	scaline
36	1	15	15	scaline
37	1	16	16	scaline
38	1	17	17	isoscles
39	1	18	18	scaline
40	15	0	0	isoscles
41	15	1	0	scaline
42	15	2	2	eqvalent
43	15	3	2	isoscles
44		4	3	isoscles
45	15	5	4	saline
46	15	6	5	scaline
47	15	7	6	scaline
48	15	8	7	isosles
49	15	9	8	ioscles
50	15	10	9	scaline
51	15	11	10	scaline
52	15	12	12	isoscles
53	15	13	12	scaline
54	15	13	13	isoscles
55	15	14	14	isoscles
56	15	15	15	scaline
57	15	16	16	eqvalent

58	15	17	17	isoscles
59	15	18	18	scaline
60	15	19	19	scaline
61	15	20	20	isoscles
62	15	21	21	isoscles
63	15	22	22	scaline
64	15	23	23	isoscles
65	31	0	0	Isosceles
66	31	31	36	isoscles
67	31	30	31	Isocles
68	31	0	0	Not a trianlgle
69	31	1	1	Not a triangle
67	31	2	2	Not a triangle
68	31	3	3	isoscles
69	31	4	4	isocles
70	31	6	6	isoscles
71	31	7	7	isolcles
72	31	8	8	isocles
73	31	9	9	isocles
74	31	10	10	isoscles
75	31	11	11	isoscles
76	31	12	12	Not a triangle
77	31	13	13	Not a triangle
78	31	14	14	Not a triangle
79	31	31	31	eqavalent

80	31	30	35	scaline
81	31	20	20	isoscles
82	31	19	19	isoscles
83	31	25	35	scaline
84	31	26	26	isoscles
85	31	27	27	isoscles
86	31	28	28	isoscles
87	31	29	29	isoscles