**Bank ATM Project**

Part 1

By Gregory Moes

# Table of Contents:

[Table of Contents: 2](#_Toc65101541)

[Statement of Independent Effort: 3](#_Toc65101542)

[Analysis: 3](#_Toc65101543)

[Pseudocode: 3](#_Toc65101544)

[Flowchart: 4](#_Toc65101545)

[Test Cases: 5](#_Toc65101546)

[Code: 5](#_Toc65101547)

[Grading Rubric: 9](#_Toc65101548)

[Terminal Screenshots: 12](#_Toc65101549)

# Statement of Independent Effort:

I, Gregory Moes, hereby certify that is my original work completed without the assistance of anyone or any outside resources.

Gregory Moes

# Analysis:

Inputs: user, password, account selection or exit, deposit amount

Outputs: customer name, account selection, deposit amount, account totals

Processes: validate user credentials, use input to make decisions after testing input for errors, perform the decided upon calculations, display results to user

Pseudocode:

1)Declare variables: customer name, user name, password, action flags, account names (savings and checking), account balances (savings and checking), total balance

2)Welcome customer to ATM and prompt for user information or exit in loop

3)Read in user name and password

4)Test that user name and password match

5)Prompt to select account type or exit action wrapped in loop

6)Prompt to select account action or exit

7)Perform selected action to selected account

8)Display the results to customer

9)Repeat until customer is done

10)Display final results and exit program

Diagram

Description automatically generatedFlowchart:

# Test Cases:

|  |  |  |  |
| --- | --- | --- | --- |
| Case Description | Input | Output | Analysis |
| Incorrect username input | r | Invalid | My program correctly loops back to allow the user a second try. |
| Incorrect password input | b | Invalid | My program correctly loops back to allow the user a third try. |
| 3rd username/ password incorrect attempt | rb | Invalid and lockout | Program terminates after failing 3 times to validate user |
| Deposit to savings | 500 | 3000 | Program correctly adds the deposit amount to the balance |
| Deposit to checking | 1000 | 1035 | Program correctly adds the deposit amount to the balance |
| Deposit to checking and savings | 500 and 1000 | 535 and 3500 | Program correctly adds the deposit amount to the balance |
| Deposit to checking and savings | 123456789 and 123456789 | 123456824 and 123459289 and 246916113 | Program correctly adds the deposit amount to the balance |
| Deposit to checking and savings | 0 and 0 | 35 and 2500 | Program correctly adds the deposit amount to the balance |
| Multiple deposits to checking | 500, 0, 9000 | 9535 | Program correctly adds the deposit amount to the balance |
| Negative deposit | -500 | -465 | Program fails to inform user of the invalid entry and debits the account |

# Code:

//Author: Greg Moes

//ATM Bank Project Part 1

//2/24/2021

//preprocessors

#include <iostream>

#include <cmath>

#include <string>

#include <iomanip>

#include <cassert>

using namespace std;

int main()//declare the main

{

//declare and initialize variables

int checkingStartBalance = 35, totalBalance, savingsStartBalance = 2500, checkingNewBalance = checkingStartBalance, savingsNewBalance = savingsStartBalance, depositAmount, totalDeposit = 0, i= 1;

string customerName = "Robert Brown", userName, password, testName = "rbrown", testPassword = "blue123";

char selection;

bool exit = false, identity = false;

do//login loop

{

cout << "Welcome to Your Bank!" << endl;//prompt user

cout << "Please login.\nUsername: ";

cin >> userName;

if (userName == testName)//username test

{

cout << "Please enter your password: ";

cin >> password;

if (password == testPassword)//password test

{

cout << "Thank you for verifying " << customerName << "." << endl;

identity = true;//set id flag true

}//end password if

else

{

i++;//increase invalid counter for password

cout << "Invalid input! You have 3 attempts. Attempt #" << i << endl;

}//end password else

}//end username if

else

{

i++;//increase invalid counter for username

cout << "Invalid input! You have 3 attempts. Attempt #" << i << endl;

}//end username else

if (i == 4)//test for lockout

assert(0);

}while (identity == false);//test for id flag

while ((identity == true) && (exit == false))//test id flag and exit flag

{

cout << "Welcome " << customerName << ".\nPlease select checking (C), savings (S), or exit (E): ";//prompt user again

cin >> selection;

switch (toupper(selection))//menu selection

{

case 'C': cout << "\nChecking Account\n";//checking account deposit

cout << "You are making a deposit to your checking account.\nPlease enter the desired deposit amount: ";

cin >> depositAmount;

checkingNewBalance += depositAmount;

totalDeposit += depositAmount;

cout << "\nYou deposited $" << depositAmount << " into your checking account.\nYour new balance is: $"<< checkingNewBalance << endl << endl;

break;

case 'S': cout << "\nSavings Account\n";//savings deposit

cout << "You are making a deposit to your savings account.\nPlease enter the desired deposit amount: ";

cin >> depositAmount;

savingsNewBalance += depositAmount;

totalDeposit += depositAmount;

cout << "\nYou deposited $" << depositAmount << " into your savings account.\nYour new balance is: $"<< savingsNewBalance << endl << endl;

break;

case 'E': cout << "\nThank you for your business.\nHave a Good Day!\n\n";//exit from menu

exit = true;

break;

default : cout << "Invalid entry!\n";//invalid entry

}

}//end while

//display results to customer

totalBalance = checkingNewBalance + savingsNewBalance;

cout << setprecision(2) << fixed << showpoint;

cout << "Thank you " << customerName << ", and here is your statement:";

cout << "\nStarting Checking Balance: $" << checkingStartBalance << "\nStarting Savings Balance: $" << savingsStartBalance << "\nStarting Total: $" << checkingStartBalance + savingsStartBalance;

cout << "\n\nTotal Deposit: $" << totalDeposit;

cout << "\n\nNew Checking Balance: $" << checkingNewBalance << "\nNew Savings Balance: $" << savingsNewBalance << "\nNew Total: $" << totalBalance << endl;

return 0;

}

# Grading Rubric:

*Fundamentals of Programming*

*Dr. Yohn Parra*

*Before submitting the project package, the student should review each of the elements listed below and put a checkmark only in those checkboxes where the designated elements has been reviewed and meets specifications. After completing your document package, number your pages and write the designated page numbers onto the spaces provided on the grading sheet.*

**\_\_\_\_\_ Professionalism (10 points)**

* Following directions
* Neatly assembled 8 ½ by 11
* Cover page
* Page numbers
* Documentation

**\_\_\_\_\_ Source Program Listing and Proper Execution of Program (30 points)**

*It is expected that each student’s program will run correctly*

* Program source code listing matches code on submission and/or backups
* Inclusion of comment lines in source code
* Comments at the beginning of the program including programmer, project name and number, date written, and brief program description.
* Comments at key locations throughout the code
* Descriptive variable names (that follow naming convention)
* Logic is correct
* Logic is clear and easy to follow
* Proper formatting of statements
* Alignment, proper indentation, etc
* Proper use of data types and data conversions

**\_\_\_\_\_ Test Data (5 points)**

* Each test case properly calculated by hand and documented
* Suitable choice of you own test data case

**\_\_\_\_\_ Input Window (10 points)**

* Correct data type for each input section
* Analysis of data type (e.g. int, float, double etc.)
* Appropriate restrictions for each input section
* Data input value shown matches specified test data
* Appropriate display for each input section

**\_\_\_\_\_ Output (15 points)**

* Suitable layout of output (including required fields, easy to read layout, etc.)
* All data cases displayed
* Correct value displayed for each case
* Correct format of fields (e.g. use of integers and not float as appropriate, dollars and cents, display of $, etc)
* Required output format
* Aesthetics (User-friendliness, easy to understand output, alignments, etc)

**\_\_\_\_\_ Documentation (40 points)**

* Analysis of specifications
* Pseudocode
* Flowchart
* Hard copy of program

**\_\_\_\_\_ Fully Functioning Program (30 points)**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Possible points = 140

Points Earned =

# Terminal Screenshots:

