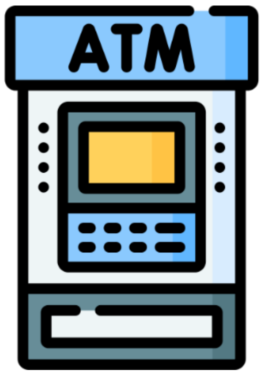
CCPROG1 Machine Project Specifications
Term 1, AY 2024 – 2025

Introduction – Cash Deposit Machine (CDM) and Automated Teller Machine (ATM) Simulator



Deposit machines in banks, often referred to as Cash Deposit Machines (CDMs) or Deposit ATMs, are self-service terminals that allow customers to deposit cash and checks directly into their bank accounts without the need for a teller. On the other hand, Automated Teller Machines or ATMs are devices that dispense cash and allow you to make other banking transactions. An ATM typically consists of a screen, a card reader, a keypad, a cash dispenser and a printer. For this Machine Project, you will create a CDM/ATM Simulator by using the knowledge, skills, and techniques that you've learned in CCPROG1.

Initial Account Number, PIN, and Initial Balance

Default Account Number: The default account number is represented by your student number. This is a value that never changes; therefore, must be declared as a constant.

Default PIN: The default pin is 123456. The PIN must always be six digits long with values between 100000 and 999999 only. Any value below or above the mentioned range shall be considered invalid.

Initial Balance: The initial balance is P1000.00.

Initial Log In

When the program is run, the user will be asked to input the account number together with the PIN. It is assumed that the kiosk offer cardless transactions.

```
---------------------------------------------------------------------
Enter Account Number: 12400001↵
Enter PIN (Six Digits): 123456↵
---------------------------------------------------------------------
```

This action may yield two results:

1. Upon successful input of a matching account number and PIN, the user will be greeted by the system and then directed to the MAIN MENU;

```
---------------------------------------------------------------------
Enter Account Number: 12400001↵
Enter PIN (Six Digits): 123456↵
Welcome, 12400001!
---------------------------------------------------------------------
```

2. Upon unsuccessful inputs of the account number and PIN, a prompt must be shown on the screen indicating that the user may input the credentials again. However, after three successive unsuccessful logins, the user will be prompted that the program will terminate.

First and second unsuccessful attempt:

```
----------------------------------------------------------------------
Enter Account Number: 12400001↲
Enter PIN (Six Digits): 654321↲
Sorry, the account number and PIN did not match. Please try again.
----------------------------------------------------------------------
```

On the third unsuccessful attempt:

```
----------------------------------------------------------------------
Enter Account Number: 12400001↲
Enter PIN (Six Digits): 654321↲
Failed logins have reached three attempts. The transaction will be
terminated!
----------------------------------------------------------------------
```

The Main Menu

The main menu will continue to be shown and transactions can be chosen indefinitely until option `[0]` is chosen. The main menu is composed of the following options:

```
----------------------------------------------------------------------
[1] – Change PIN
[2] – Balance Inquiry
[3] – Deposit
[4] – Withdraw
[5] – Transfer Funds
[6] – Pay Bills
[7] – View Transactions
[0] – Logout
Enter option number:
----------------------------------------------------------------------
```

[1] – Change PIN

Choosing this option prompts the user to change the associated PIN with the account. This option requires the user to re-enter the current PIN. If the input is correct, the user will be asked to nominate a new PIN with the following restrictions:

1. The new PIN must not be the same with the current PIN;
2. The new PIN must be typed in twice and required to be of the same value. Upon successful PIN change, the user will be asked to re-login.

Correct current PIN and new PIN inputs

```
----------------------------------------------------------------------
Enter current PIN (Six Digits [100000 - 999999]):123456↵
Enter new PIN (Six Digits [100000 - 999999]): 999999↵
Verify new PIN: 999999↵
PIN changed successfully. Please re-login.
----------------------------------------------------------------------
```

If the input is wrong, the user is asked to retype the PIN. After the third unsuccessful attempt, the transaction will be terminated and the account will be logged out.

[2] – Balance Inquiry

This option is very straightforward. This will simply show the current balance of the account.

```
----------------------------------------------------------------------
Account Number: 12400001
Your current balance is: P1000.00
----------------------------------------------------------------------
```

[3] – Deposit

Depositing money into the machine is done virtually. This means that we only assume that the user is placing actual money into the machine. Once chosen, the user will be presented with the following interface:

```
----------------------------------------------------------------------
Choose your deposit denomination:
[1] - P100
[2] - P200
[3] - P500
[4] - P1000
[0] - Cancel
Denomination: 1↵
Number of bills (up to 10 bills only): 5↵
Current Deposit: P500.00
Running Deposit: P500.00
Do you wish to deposit more? Y/N: N↵
----------------------------------------------------------------------
```

Denomination

The user may choose among the four valid denominations. If the user entered an incorrect option, he will be asked to input again until the correct option is chosen. Should the user choose '0', the deposit transaction will be terminated.

Number of Bills

The machine can only process up one to ten bill per denomination per transaction. A value of zero or eleven or more is not valid.

More Deposits

If the user wishes to deposit more, a prompt of ‘Y’ or ‘y’ will allow the user to repeat the deposit process. The transaction will be terminated if the user inputs any other character to the last prompt or ‘0’ to the denomination prompt.

Current Deposit

This value corresponds to the current deposit on the live transaction. This is the chosen denomination multiplied by the number of bills for the current transaction.

Running Deposit

This value corresponds to the sum of all the preceding deposits before the user ends the denomination prompt.

[4] – Withdrawal

This feature allows the user to withdraw money from the account with the following restrictions:

1. The withdrawal is not allowed if the balance is P0.00;
2. The withdrawal amount must always be greater than the current balance;
3. The withdrawal amount cannot be a negative number;
4. The withdrawal amount must be divisible by 100;
5. A withdrawal amount of P0.00 will cancel the transaction.

Successful withdrawal

```
----------------------------------------------------------------------
How much do you wish to withdraw? Input '0' to cancel (Intervals of P100
only): 300↲
P300.00 has been withdrawn.
----------------------------------------------------------------------
```

Balance is insufficient

```
----------------------------------------------------------------------
How much do you wish to withdraw? Input '0' to cancel (Intervals of P100
only): 1100↲
Sorry, your balance in insufficient. Your current balance is P1000.00.
----------------------------------------------------------------------
```

Withdrawal amount is not divisible by 100

```
----------------------------------------------------------------------
How much do you wish to withdraw? Input '0' to cancel (Intervals of P100
only): 50↲
You can only withdraw by intervals of P100.00.
----------------------------------------------------------------------
```

Withdrawal amount is negative

```
----------------------------------------------------------------------
How much do you wish to withdraw? Input '0' to cancel (Intervals of P100
only): -100↵
You can only withdraw a negative amount.
----------------------------------------------------------------------
```

Withdrawal amount is zero

```
----------------------------------------------------------------------
How much do you wish to withdraw? Input '0' to cancel (Intervals of P100
only): 0↵
Transaction is cancelled.
----------------------------------------------------------------------
```

[5] – Transfer of Funds

This feature allows the user to transfer money from the originating account to any account. The following specifications and restrictions must be observed:

1. The account number to be transferred to must be exactly eight digits from 10000000 to 99999999. Any input that is outside this range will be asked to be re-entered;
2. The amount to be transferred must not be greater than the current balance of the account;
3. The transfer amount must be P1.00 to P50000.00 only;
4. An input of zero will cancel the transaction.

Successful transfer of funds

```
----------------------------------------------------------------------
Enter Account Number (Eight Digits [10000000 - 99999999]) or 0 to cancel:
12345678↵
Enter amount to transfer or 0 to cancel: 100
100.00 has been transferred to 12345678.
----------------------------------------------------------------------
```

Input of invalid account number

```
----------------------------------------------------------------------
Enter Account Number (Eight Digits [10000000 - 99999999]) or 0 to cancel:
1234567↵
Account number must be eight digits and within the prescribed range.
Please re-enter the account number.
----------------------------------------------------------------------
```

Amount is not within the range

```
----------------------------------------------------------------------
Enter Account Number (Eight Digits [10000000 - 99999999]) or 0 to cancel:
12345678↵
Enter amount to transfer or 0 to cancel: -100↵
Transfer amount must be a positive value between 1 and 50000! Please re-
enter.
----------------------------------------------------------------------
```

Transfer Cancellation

```
----------------------------------------------------------------------
Enter Account Number (Eight Digits [10000000 - 99999999]) or 0 to cancel:
0↵
Transaction cancelled. Going back to the main menu.
----------------------------------------------------------------------
```

[6] – Pay Bills

This feature allows the user to pay a biller from the five given in the list. The user may choose a biller based on the indicated number or input zero to cancel the transaction. The following are the specifications and restrictions:

1. Only the billers listed can be paid;
2. The amount to be paid must not be greater than the current balance;
3. The payment amount must be greater than zero;
4. An input of zero cancels the transaction.

Successful bills payment

```
----------------------------------------------------------------------
[1] - Biller 1
[2] - Biller 2
[3] - Biller 3
[4] - Biller 4
[5] - Biller 5
[0] - Cancel
Choose biller: 1↵
Enter amount to pay: 100↵
100.00 has been paid to Biller 1.
----------------------------------------------------------------------
```

Invalid biller option

```
----------------------------------------------------------------------
[1] - Biller 1
[2] - Biller 2
[3] - Biller 3
[4] - Biller 4
[5] - Biller 5
[0] - Cancel
Choose biller: 6↵
The chosen biller is not available. Please choose again.
----------------------------------------------------------------------
```

Invalid payment amount

```
----------------------------------------------------------------------
[1] - Biller 1
[2] - Biller 2
[3] - Biller 3
[4] - Biller 4
[5] - Biller 5
[0] - Cancel
Choose biller: 1↵
Enter amount to pay: 0↵
Payment cannot be zero or less!
----------------------------------------------------------------------
```

[7] – View Transactions

This allows the user to view up to five of the most recent successful transactions. If the number of transactions exceed five, the oldest will be overridden. Take note that the number of transactions shown should not exceed the total number of actual transactions. For example, transactions two onwards should not be shown as "blank" if there is only one successful transaction in record. The transactions should show the following details:

1. Transaction number from 1 to 5;
2. Transaction type: Deposit, Withdrawal, Transfer of Funds, Bills Payment
3. Transaction Amount
4. Running Balance

Transactions

```
----------------------------------------------------------------------
Transaction #    Transaction      Type Amount     Running Balance

1               Deposit           +P10000.00       P11000.00
2               Withdrawal        -   P200.00      P10800.00
3               Transfer of Funds - P1500.00        P9300.00
4               Bills Payment     -   P250.00       P9050.00
5               Deposit           + P5000.00       P14050.00
----------------------------------------------------------------------
```

Other Project Specifications

1. Format your output based on your preference. Just ensure that the output is presented well. You may clear your screen or use pauses or sleep to enhance the user experience;
2. The sample outputs are just recommended. Everyone is encouraged to write their own versions of the output to ensure that there is a variety among the submissions;
3. Other concepts which are not mentioned but are considered necessary may be included in this project.

How to Approach the Machine Project

Step 1: Problem analysis and algorithm formulation

Read the MP Specifications again! Identify clearly what are the required information from the user, what kind of processes are needed, and what will be the output(s) of your program. Clarify with your professor any issues that you might have regarding the machine project.

When you have all the necessary information, identify the necessary functions that you will need to modularize the project. Identify the required data of these functions and what kind of data they will return to the caller. Write your algorithm for each of these modules/functions as well as the algorithm for your main program.

Step 2: Implementation

In this step, you are to translate your algorithm into proper C statements. While implementing, you are to perform the other phases of program planning and design (discussed in the other steps below) together with this step.

Follow the coding standard indicated in the course notes (Modules section in AnimoSpace).

You may choose to type your program in a text editor or an IDE (i.e. Dev-C IDE) at this point. Note that you are expected to use statements taught in class. You can explore other libraries and functions in C as long as you can clearly explain how this work. You may also use arrays, should these be applicable and you are able to properly justify and explain your implementation using these. For topics not covered, it is left to the student to read ahead, research, and explore by himself.

Note though that you are **NOT ALLOWED** to do the following:

- to declare and use global variables (i.e., variables declared outside any function),
- to use `goto` statements (i.e., to jump from code segments to code segments),
- to use the `break` statement to exit a block other than `switch` blocks,
- to use the `return` statement or `exit` statement to **prematurely** terminate a loop or function or program,
- to use the `exit` statement to **prematurely** terminate a loop or to terminate the function or program, and
- to call the `main()` function to repeat the process instead of using loops.

It is best that you perform your coding "incrementally." This means:

- dividing the program specification into subproblems, and solving each problem separately according to your algorithm;
- coding the solutions to the subproblems one at a time. Once you're done coding the solution for one subproblem, apply testing and debugging.

Documentation

While coding, you have to include internal documentation in your programs. You are expected to have the following:

- File comments or Introductory comments
- Function comments
- In-line comments

Introductory comments are found at the very beginning of your program before the preprocessor directives. Follow the format shown below. Note that items in between **< >** should be replaced with the proper information. Items in between ***[ ]*** are optional, indicate if applicable.

```
/*
     Description: <Describe what this program does briefly>
     Programmed by: <your name here> <section>
     Last modified: <date when last revision was made>
     Version: <version number>
     [Acknowledgements: <list of sites or borrowed libraries and
sources>]
*/
<Preprocessor directives>
<function implementation>
int main()
{
     return 0;
}
```

Function comments precede the function header. These are used to describe what the function does and the intentions of each parameter and what is being returned, if any. If applicable, include pre-conditions as well. Pre-conditions refer to the assumed state of the parameters. Follow the format below when writing function comments:

```
/*   <Description of function>
     Precondition: <precondition / assumption>
     @param <name> <purpose>
     @return <description of returned result>
*/
<return type> <function name> (<parameter list>);
```

Example:

```
/* This function computes for the area of a triangle
   Precondition: base and height are non-negative values
   @param base is the base measurement of the triangle in cm
   @param height is the height measurement of the triangle in
   cm
   @return the resulting area of the triangle
*/
float
getAreaTri (float base, float height)
{
    ...
}
```

In-Line comments are other comments in major parts of the code. These are expected to explain the purpose or algorithm of groups of related code, esp. for long functions.

Step 3: Testing and Debugging

SUBMIT THE LIST OF TEST CASES YOU HAVE USED.

For each feature of your program, you have to fully test it before moving to the next feature. Sample questions that you should ask yourself are:

1) What should be displayed on the screen after a user input?
2) What would happen if inputs are incorrect? (e.g., values not within the range)
3) Is my program displaying the correct output?
4) Is my program following the correct sequence of events (correct program flow)?
5) Is my program terminating (ending/exiting) correctly? Does it exit when I press the command to quit? Does it exit when the program's goal has been met? Is there an infinite loop?
6) and others...

IMPORTANT POINTS TO REMEMBER:

1. You are required to implement the project using the C language (C99 and NOT C++). Make sure you know how to compile and run in both the IDE (DEV-C++) and the command prompt (via
   `gcc -Wall -std=c99 <yourMP.c> -o <yourExe.exe>`
2. The implementation will require you to:
   a. Create and Use Functions
   b. **Note:** Non-use of self-defined functions will merit a grade of **0** for the **machine project**. Too few self-defined functions may merit deductions. A general rule is to create a separate function for each option described above, unless some features are too similar that one function can serve the purpose for two [or more] of the options. Note that functions whose tasks are only to display are not included in the count for creating user-defined functions.

    c. Appropriately use conditional statements, loops and other constructs discussed in class (Do not use brute force solution. **You are not allowed to use goto label statements, exit statements. You are required to pass parameters to functions and not allowed to declare global or static variables.**) Refer to Step 2 on Implementation for other details and restrictions.
        i. Consistently employ coding conventions
    d. Include internal documentation (i.e., comments)

3. Deadline for the project is **7:59AM of November 25, 2024 (Monday)** via submission through **AnimoSpace**. Late submission up to max of 3 days will incur deductions. That is, submissions from 8:00AM of November 25 to 7:59AM of November 26 will incur 10% deduction, submissions from 8:00AM of November 26 to 7:59AM of November 27 will incur additional 20% deduction (total of 30% deduction), submissions from 8:00AM of November 27 to 7:59AM of November 28 will incur additional 30% deduction (total of 60% deduction in MP score), and submissions from 8:00AM of November 28 will not be accepted anymore as facility is locked. Once locked, no MP will be accepted anymore and this will result to a **0.0** for your machine project.
4. The following are the deliverables:

Checklist:

- Upload in **AnimoSpace** by clicking **Submit Assignment** on Machine Project and adding the following files:
    - source code*
    - test script**
- email the softcopies of everything as attachments to YOUR own email address on or before the deadline

Legend:

* Source Code also includes the internal documentation. The **first few lines of the source code** should have the following declaration (in comment) **BEFORE** the introductory comment:

```
/*************************************************************
******
This is to certify that this project is my own work, based on
my personal efforts in studying and applying the concepts
learned. I have constructed the functions and their respective
algorithms
and corresponding code by myself. The program was run, tested,
and debugged by my own efforts. I further certify that I have
not copied in part or whole or otherwise plagiarized the work
of other students and/or persons.

<your full name>, DLSU ID# <number>
*************************************************************
*****/
```

** Test Script should be in a table format, with header as shown below. There should be at least 3 distinct test classes (as indicated in the description) **per function**. There is no need to test functions which are only for screen design.

Function Name	**#**	**Test Description**	**Sample Input** *(either from the user or passed to the function)*	**Expected Result**	**Actual Result**	**P/F**
`getAreaTri()`	1	base and height measurements are less than 1	base = 0.25 height = 0.75	...	...	...
	2					
	3					

5. **MP Demo:** You will demonstrate your project on a specified schedule during the last weeks of classes. Being unable to show up on time during the demo or being unable to answer convincingly the questions during the demo will merit a grade of 0.0 for the **MP**. The project is initially evaluated

via black box testing (i.e., based on output of running program). Thus, if the program does not compile successfully using `gcc -Wall -std=c99` and execute in the command prompt, a grade of 0 for the project will be incurred. However, a fully working project does not ensure a perfect grade, as the implementation (i.e., correctness and compliance in code) is still checked.

6. Any requirement not fully implemented and instruction not followed will merit deductions.
7. This is an **individual project**. Working in collaboration, asking other people's help, borrowing or copying other people's work or from books or online sources (either in full or in part) are considered cheating. Cheating is punishable by a grade of **0.0** for **CCPROG1** course. Aside from which, a cheating case may be filed with the Discipline Office.
8. **Bonus points:** You can earn up to **at most 10 points** bonus for additional features that your program may have. Some of the indicated additional features may require self-study. Also, any additional feature not stated here may be added but **should not conflict with whatever instruction was given in the project specifications**. Bonus points are given upon the discretion of the teacher, based on the difficulty and applicability of the feature to the program. Note that **bonus points can only be credited if all the basic requirements are fully met** (i.e., complete and no bugs). Sample bonus features may include the masking of the account number and PIN with '*' or any other character.

HONESTY POLICY AND INTELLECTUAL PROPERTY RIGHTS

Honesty policy applies. Please take note that you are NOT allowed to borrow and/or copy-and-paste – in full or in part any existing related program code from the internet or other sources (such as printed materials like books, or source codes by other people that are not online). **You should develop your own codes from scratch by yourself.**