Assignment 1

Pedram Pasandide

Due Date: 14 July

1. You are tasked with creating a program that simulates a simple banking system.

   a. (**5 points**). The system should allow users to deposit and withdraw funds from their saving accounts. Each account has a balance represented as a **floating-point number**. Your program should prompt the user for their initial balance using a `printf` and `scanf` and allow them to perform operations, including:

   - **Deposit**: Prompt the user for an amount to deposit and add it to their account balance.
   - **Withdraw**: Prompt the user for an amount to withdraw and subtract it from their account balance.
   - **Check Balance**: Display the current balance of the account.

   The program should continue to prompt the user for operations until they choose to exit. Every time you have to print out the result! To do this you can print out four options like:

   ```
   Select an operation:
   1. Deposit
   2. Withdraw
   3. Check Balance
   0. exit
   ```

   Then use another `printf` asking the user to input the option followed by a `scanf(" %<the placeholder for the variable>", &choice);` to receive the option with the proper placeholder. Using a `switch` for comparison inside a `do` or `while` loop would help you to create an infinite loop giving mentioned options until the condition of the loop is not satisfied (choosing `0` to exit).

   Implement the mentioned C code and store them in a file with name `ATM_Bank_a.c`.

   b. (**5 points**). Extend the banking system program from **Part a** to include **interest calculation**. The simplest definition of interest rate is a percentage that will be added to your saving account balance after **a year**. For example, if I have 1000$ in my back account with 0.15 (or 15%) interest rate, after a year I will have $1000 \times (1+0.15) = 1150$

in my saving bank account. The general formulation to calculate the future balance after multiple years is:

$$FutureBalance = CurrentBalance \times (1 + IR)^{years}$$

Where $years$ is the number of years that the balance is kept in the account with an interest rate of IR. To calculate $(1 + IR)^{years}$ you need to use `math.h` library.

The interest rate should be constant and defined as a **floating-point number**. Prompt the user to enter their desired interest rate **when the program starts** (We know that in the real world the interest rate is determined by the Bank! But here user must define it!)

So, your program should introduce a new operation:

- **Future Balance**: Use the mentioned equation to calculate the the future balance

So by now you have to print out possible options like:

```
Select an operation:
1. Deposit
2. Withdraw
3. Check Balance
4. Future Balance Using Interest Rate
0. exit
```

But don't forget **if** the user chooses option `4`, inside the `switch` it must prompts user to a message asking for the number of `years` and after the calculation, print out the result.

The program should continue to prompt the user for operations until they choose to exit, the same as **Part a**.

Implement the mentioned C code and store them in a file with name `ATM_Bank_b.c`.

c. (**5 points**). Expand the banking system program from **Part b** to handle **multiple accounts**. Implement an **array** of account balances, with **each element** representing an account. Prompt the user to enter the number of accounts **when the program starts** (You may leave this part for now, until Monday's lecture, 14 July, when we talk more about arrays but here I give you some hints).

Your program should introduce the following operations:

- **Switch Account**: Prompt the user for an account number and allow them to switch to a different account.
- **Display All Balances**: Display the current balances of all accounts.

By now user must be able to see following options in the terminal:

```
Current Account: <here the number of current account must be printed>
Select an operation:
1. Deposit
2. Withdraw
3. Check Balance
4. Future Balance Using Interest Rate
5. Switch Account
6. Display All Balances
0. exit
```

Define an array using `<type> balances[DEFAULT_ACCOUNTS]`, where `<type>` is the type of all numbers saved in the array, which in this case it must be a floating-point precision. `balances` is the name given to your array, you can pick anything. So we are declaring an array of variables without initializing the values. `DEFAULT_ACCOUNTS` is the number of values that by default `balances` array can keep. You must define `DEFAULT_ACCOUNTS` like the following format:

```
<including necessary library(s)>

#define MAX_ACCOUNTS 10

int main()
{
 <type> balances[DEFAULT_ACCOUNTS];
 <your code>
}
```

So in this case, the default value for `DEFAULT_ACCOUNTS` is equal to `10`. Then inside the `int main()` function before every thing, ask user the number of accounts using `scanf`. Then inside a `for` loop ask for the each balance using
`scanf("%<put the proper placeholder>", &balances[i])`
Where `i` is the counter for the `for` loop.

For example, if we have 3 accounts with 1000$, 2000$, and 1500$ balances,

balances[0] = 1000

balances[1] = 2000

balances[2] = 1500

A reminder, indexing in C starts from zero, that's why the first element of array `balances` can be accessed by `balances[0]`.

The program should continue to prompt the user for operations until they choose to exit!

Implement the mentioned C code and store them in a file with name `ATM_Bank_c.c`.

Submit On Avenue to Learn following files:

- `ATM_Bank_a.c`, `ATM_Bank_b.c`, and `ATM_Bank_c.c`
- A short description saying how the code must be compiled and executed to get the result.

Looking for Perfect Score?! A perfect submission must:

- see the potential mistakes made by user and print out a message mentioning why the input is not acceptable. So your code must be able to handle errors instead of crashing or making mistakes in calculations. We don't see an ATM in a bank giving us a C language error! Let's say to withdraw or deposit, the received number must be positive!
- use proper types of variable. If there is somewhere that you have asked to save a number, you have to make sure that you are using the proper type of variable to save the number. Let's say, does it have digits after decimal point? if it does how many digits matters?
- submit all the requested files!