Rutgers University
Program Methodology I
Spring 2018

Project 1 - Instructions

A. Functions

In this project, you need the following essential functions:

- **menu()** - this function displays all the available options that a bank clerk can perform. The menu should be displayed at first and after completion of a task. The format as follow:

```
Welcome! Select options below:
      1. Make new account.
      2. Deposit to an account.
      3. Withdraw from an account.
      4. Transfer Money from one account to another.
      5. Print account.
      6. Activate/Deactivate an account.
      7. Delete an account.
      8. Display all accounts.
      9. Quit.
Selection:
```

- **makeAccount()** - this function create the new bank account. The bank account has account number, first name, last name, and account balance. The account number is a 4-digit number. It can be generated randomly. Two bank accounts cannot have the same account number. When generating the new bank account number, make sure that it is not existed in the current list of bank accounts. The user can input first name, last name, and the starting balance. The starting balance is a double number. The format as follow:

```
Creating bank account number 9205
Enter first name: John
Enter last name: Smith
Enter starting balance: 320.56
```

- **printAllAccounts()** - this function print out all the current accounts. The format as follow:

```
Account number: 9119        Balance: 124.56
      Last name: Doe        First name: John

Account number: 9205        Balance: 320.56
      Last name: Smith            First name: John

Account number: 9305        Balance: 562.34
      Last name: Jane       First name: Mary
```

- **printAccount()** - this function print out only one account. The function prompts for the account number from the user. If the account exists, the function print out the bank account with the format as before. If the account does not exist, the function print out the a prompt that the account does not exist.

- **Transfer()** - this function transfers a certain amount of money from one account to another. It prompts for sender's account number, receiver's account number and amount of money to be transferred. The function should check whether the account numbers are valid and make sure sender has enough money for the transfer.

```
Enter account number for the sender: 9119
Enter account number for the receiver:1251
Enter amount to be transferred: 47.5
```

- **depositAccount()** - this function deposit an amount to the bank account. The function prompt the user for bank account and balance to be deposited. After the deposit, the bank account should be updated with the balance. You can use the print out function to check this. If the account does not exist, display a message that the account does not exist. The format as follow:

```
Enter account number for deposit: 9119 Enter
amount to be deposited: 23.45
```

- **ActiveDeactive()** - this function activates/deactivates an account given its account number. If an account is no active, it cannot be modified in any way through other functions. In other words, if a not active account is passed to other functions, they should show an error to the user that the process cannot be done because the account is not active. Furthermore, we assume an account is active at first unless it is deactivated using this function.

- **withdrawAccount()** - this function withdraw an amount from the bank account. The function prompts the user for bank account and balance to be withdrawn. After the withdrawal, the bank account should be updated with the balance. You can use the print out function to check this. If the account does not exist, display a message that the account does not exist. The format as follow:

```
Enter account number for withdrawal: 9119
Enter amount to be withdrawn: 56.78
```

- **deleteAccount()** - this function delete an account. The function prompts the user for the account number to be deleted. After deletion, the list of bank accounts should be updated. You can use the the print out function to check this. If the account does not exist, display a message that the account does not exist. The format as follow:

```
Enter account number to be deleted: 9119
```

- **sortAccounts()** - this function sort the bank account by the account numbers. Use this function after the **makeAccount()** function to sort the accounts after making new account.

- **validInput()** - this is a general function. This function check if the user input is valid. This function only checks the input number. The number could be the account number (for printing, deposit, or withdraw an account), or amount input (starting balance, deposit, or withdraw). Write this function in a way that it can be reused for all the function that needs inputting numbers. This function should be used with **makeAccount()**, **printAccount()**, **depositAccount()**, **withdrawAccount()**, and **deleteAccount()** functions. The followings are the few examples.

    - <u>Creating new account:</u>

```
Creating bank account number 2717 Enter
first name: Jane
Enter last name: Smith
Enter starting balance: -12 Enter

positive number: ewq

Invalid input.
Enter new number: 12
```

- Deposit to an account (1):

```
Enter account number for deposit: 2717
Enter amount to be deposit: -12

Enter positive number: rew

Invalid input.
Enter new number: 12
```

- Deposit to an account (2):

```
Enter account number for deposit: eqwr

Invalid input.
Enter new number: 1234
Account number not found.
```

B. Program Structure

Create a **vector** of the bank accounts. Each element of the vector will have all the account information. To do this, you would have to use data structure. The keyword **struct** in C++ will make a data structure. Inside the data structure, you will define all of the account information. The following is an example:

```
struct Account{
      int accountNumber;
      string lastName;
      string firstName;
      double accountBalance;
      bool active;
};

vector <Account> bankAccounts;
```

The code snippet above defines **data structure type Account** and a **vector name bankAccounts** of **type Account**. Each element of the vector will have member accountNumber, lastName, firstName, accountBalance and active (whether the account is active or not).

The vector `bankAccounts` should be **defined in the main() function**. This means that as long as the program running (the bank is operating), we have the list of bank accounts. The program should be running until terminated by the user (until he/she presses 9 in the menu).

In order to update the information of bank accounts in the vector, you would have to pass the `bankAccounts` variable **by reference** to the functions. Therefore, in the definition of all the functions you have an argument like `Account` in the following code:

```
void printAllAccounts (vector<account>& Accounts);
```

then you will be able to pass the `bankAccounts` variable to the functions as follows:

```
printAllAccounts(bankAccounts);
```

Happy coding!