HW1: Bank Accounts:

You have been hired as a programmer by a major bank. Your first project is a small banking transaction system. Each account consists of a number and a balance. The user of the program (the teller) can create a new account, as well as perform deposits, withdrawals, balance inquiries, etc..

Initially, the account information of existing customers is to be read into a pair of parallel arrays--one for account numbers, the other for balances. The bank can handle up to **MAX_NUM** accounts. Use the following function to read in the data values:

```
int read_accts(int acctnum_array[], double balance_array[], int max_accts);
```

This function fills up the account number and balance arrays (up to **max_accts**) and returns the actual number of accounts read in (later referred to as **num_accts**).

After initialization, **print the initial database of accounts and balances**. Use function print_accts() described below.

The program then allows the user to select from the following menu of transactions:

Select one of the following:
- W - Withdrawal
- D - Deposit
- N - New account
- B - Balance
- Q – Quit
- X – Delete Account

Use the following function to produce the menu:

```
void menu();
```

This function only displays the menu. The **main program** then prompts the user for a selection. You should verify that the user has typed in a valid selection (otherwise print out an error message and repeat the prompt).

Once the user has entered a selection, one of the following functions should be called to perform the specific transaction. **At the end, before the user quits, the program prints the contents of the account arrays**.

```
int findacct(int acctnum_array[], int num_accts, int requested_account);
```

This function **returns the index of requested_account in acctnum_array** if the account exists, and -1 if it doesn't. It is called by all the remaining functions.

```
void withdrawal(int acctnum_array[], double balance_array[], int num_accts);
```

This function prompts the user for the account number. If the account does not exist, it prints an error message. Otherwise, it asks the user for the amount of the withdrawal. If the account does not contain sufficient funds, an it prints an error message and does not perform the transaction.

```
void deposit(int acctnum_array[], double balance_array[], int num_accts);
```

This function prompts the user for the account number. If the account does not exist, it prints an error message. Otherwise, it asks the user for the amount of the deposit.

```
int new_acct(int acctnum_array[], double balance_array[], int num_accts);
```

This function prompts the user for a new account number. If the account already exists, it prints an error message. Otherwise, it adds the account to the account array with an initial balance of 0. It returns the new number of accounts.

```
int delete_acct(int acctnum_array[], double balance_array[], int num_accts);
```

This function prompts the user for an account number. If the account does not exist, or if the account exists but has a non-zero balance, it prints an error message. Otherwise, it deletes the account. It returns the new number of accounts.

```
void balance(int acctnum_array[], double balance_array[], int num_accts);
```

This function prompts the user for an account number. If the account does not exist, it prints an error message. Otherwise, it prints the account balance.

```
void print_accts(int acctnum_array[], double balance_array[], int num_accts);
```

This function prints all customer information--account number and balance.

Notes:

1. All output must be file directed

2. Only output must go to the file - not interactive prompts or menus (which go to the monitor).

3. No global variables are allowed

4. The program and all functions must be properly commented.

5. The program must be properly tested.

- **a. The initial database should consist of at least 5 accounts**
- **b. The initial database should be printed to the file**
- **c. Test at least 2 invalid menu selections**
- **d. Test at least 2 balance inquiries:**
  - **i. valid account**
  - **ii. invalid account**
- **e. Test at least 3 deposits:**
  - **i. valid account - valid deposit amount**
  - **ii. valid account - invalid deposit amount**
  - **iii. invalid account**
- **f. Test at least 4 withdrawals:**
  - **i. valid account - valid withdrawal amount**
  - **ii. valid account - invalid withdrawal amount**
  - **iii. valid account - insufficient funds**
  - **iv. invalid account**
- **g. Create at least 3 new accounts with an initial balance of 0.0**
- **h, Test the creation of at least 1 invalid new account**
- **i. Test several transactions on the new accounts (deposits, withdrawals, etc.)**
- **j. Delete at least two accounts (1 old account and 1 new account but not the last new account created)**
- **k. quit and print the final database**