Problem B: Bank Account Transactions Using Random Access Files

(30% related to Lab 14)

Problem Description

Modify the code of the function **void newRecord(fstream &insertInFile)** in Fig. 17.7 into **int newRecord(fstream &insertInFile)**. The modified function should automatically find an account number that is not used for creating a new record. The valid account number is from 1 to 100. The distance between two adjacent **used** accounts B and C where no other used accounts are between them is defined as ***Dist*(B,C) = *abs*(account number of B – account number of C)** where *abs*(…) returns the absolute value of the argument. In order to separate the accounts far apart from each other, the account number for insertion should be the one equal to **max Dist(B, C)/2 + min(B,C)** for all possible B and C. Here, we assume that account 0 and account 101 are used. They cannot be used as customer's accounts. Hence, line 162 in page 741 should be replaced by a function called **int getFarApartUnusedActNo(fstream &)**. This function returns the far apart account number. If there is not any unused account number, it should return -1. Moreover, if a new account can be created successfully, the modified function should return the account number. Basically, you have to rewrite newRecord(…) because of using getFarApartUnusedActNo(…). You must also implement a new function **void printAllRecords(fstream &)** that will print out all the valid records, i.e., printing only the record whose account number is greater than zero. This function should make a call to outputLine() to print out a record (account). The given main() function will perform a series of transactions. The account information is stored in a file called *credit.dat* which is a random access file (binary file). A transaction will typically modify the file content. The correctness of file content should be maintained after each transaction.

Input format

The input to the program will be the data used for performing transactions. Each input line consists of data used for a transaction, which depends on the task done for a transaction.

Output format

The output will be a modification to the file *credit.dat* which should be printed out after all transactions are done. Refer to the example output for the output format.

Requirements

You have to use the header file given in Fig. 17.2. The main() function has been modified and given to you below. You should not modify the given main() here.

```
int main()
{
    // open file for reading and writing
    fstream inOutCredit( "credit.dat", ios::in | ios::out | ios::binary );

    // exit program if fstream cannot open file
    if ( !inOutCredit )
    {
        cerr << "File could not be opened." << endl;
        exit ( 1 );
    } // end if

    int choice; // store user choice
```

```
     printAllRecords(inOutCredit);
     inOutCredit.clear(); // reset end-of-file indicator
    // enable user to specify action
    while ( ( choice = enterChoice() ) != END )
    {
        switch ( choice )
        {
            case PRINT: // create text file from record file
                createTextFile( inOutCredit );
                break;
            case UPDATE: // update record
                updateRecord( inOutCredit );
                break;
            case NEW: // create record
                newRecord( inOutCredit );
                break;
            case DELETE: // delete existing record
                deleteRecord( inOutCredit );
                break;
            default: // display error if user does not select valid choice
                cerr << "Incorrect choice" << endl;
                break;
        } // end switch

        inOutCredit.clear(); // reset end-of-file indicator
    } // end while
    printAllRecords(inOutCredit);
} // end main
```

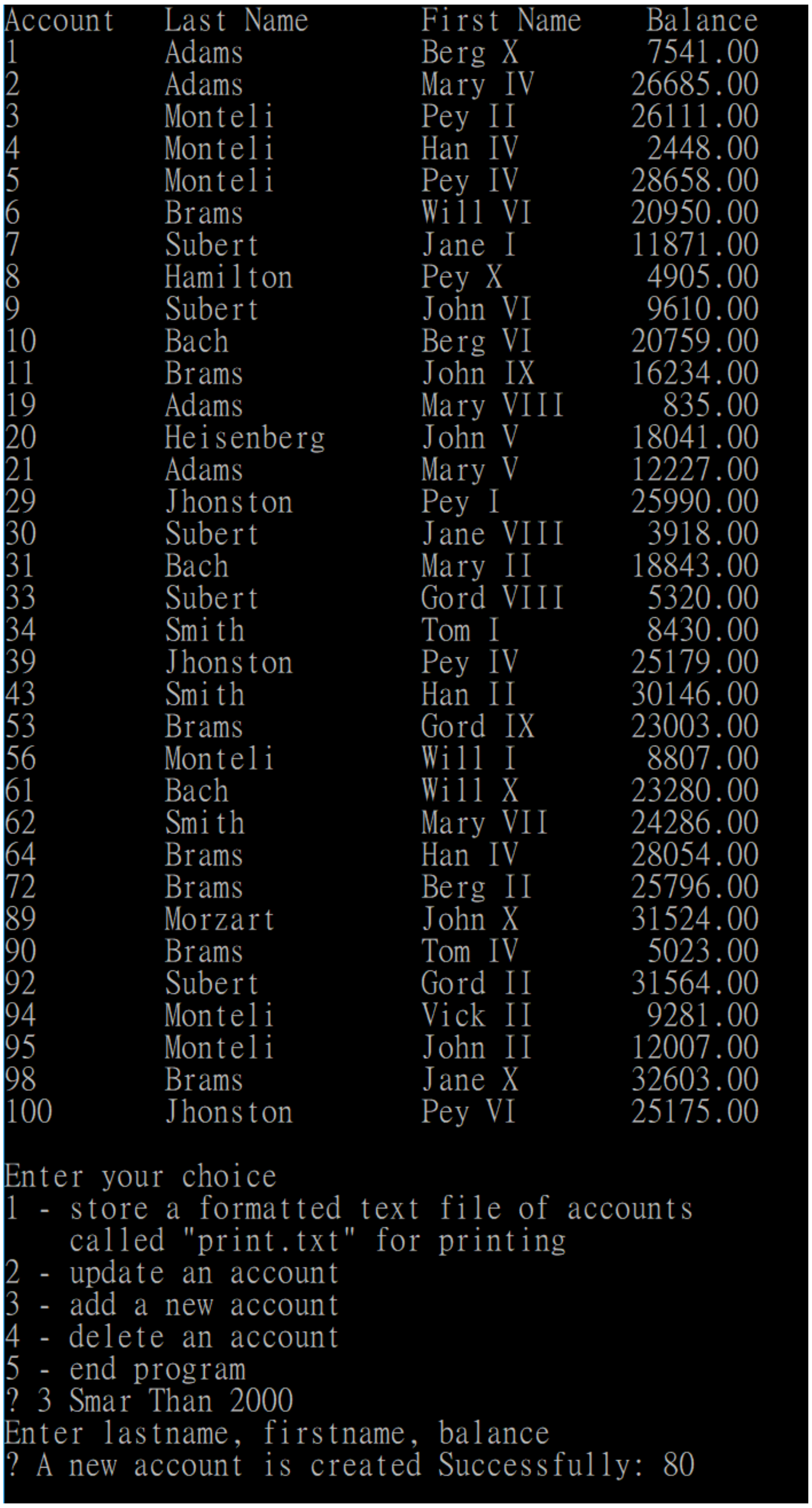
Hints

Whenever applying seekp or seekg to move the file position pointer, you must make sure that **eof** flag should be cleared (i.e., should not be set).

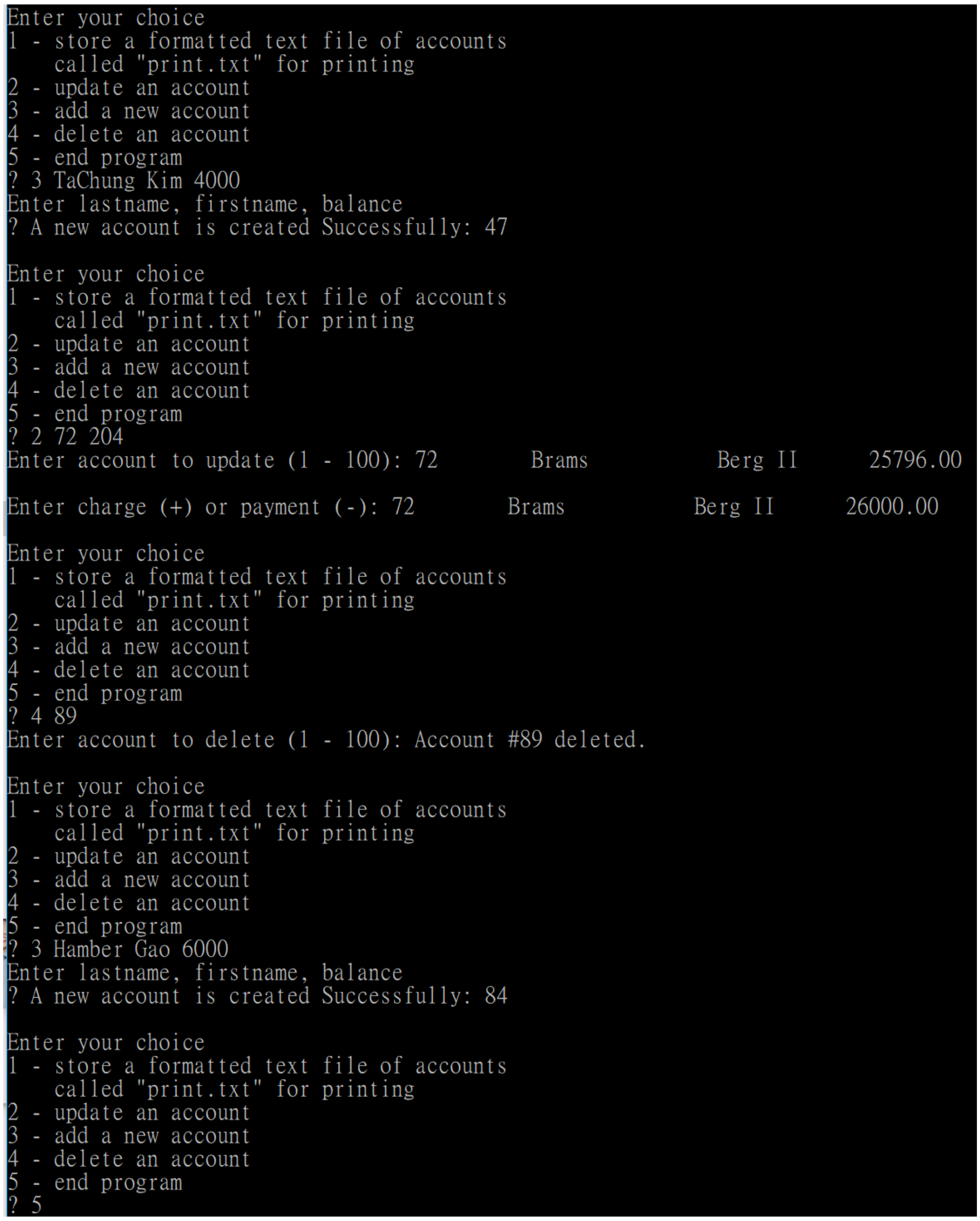
Example Input:

```
3 Smar Than 2000
3 TaChung Kim 4000
2 72 204
4 89
3 Hamber Gao 6000
5
```

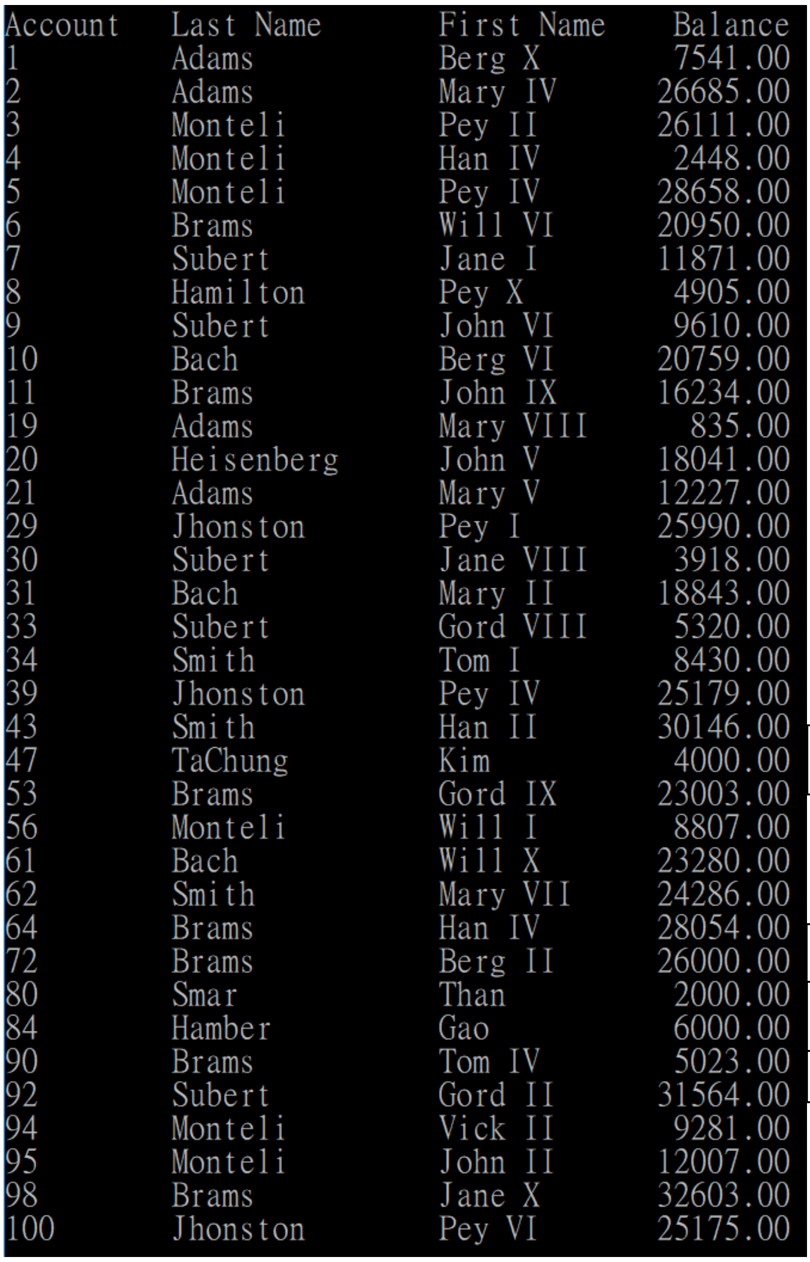
Example Output (containing input):



```
Account    Last Name       First Name     Balance
1          Adams           Berg X         7541.00
2          Adams           Mary IV       26685.00
3          Monteli         Pey II        26111.00
4          Monteli         Han IV         2448.00
5          Monteli         Pey IV        28658.00
6          Brams           Will VI       20950.00
7          Subert          Jane I        11871.00
8          Hamilton        Pey X          4905.00
9          Subert          John VI        9610.00
10         Bach            Berg VI       20759.00
11         Brams           John IX       16234.00
19         Adams           Mary VIII       835.00
20         Heisenberg      John V        18041.00
21         Adams           Mary V        12227.00
29         Jhonston        Pey I         25990.00
30         Subert          Jane VIII      3918.00
31         Bach            Mary II       18843.00
33         Subert          Gord VIII      5320.00
34         Smith           Tom I          8430.00
39         Jhonston        Pey IV        25179.00
43         Smith           Han II        30146.00
53         Brams           Gord IX       23003.00
56         Monteli         Will I         8807.00
61         Bach            Will X        23280.00
62         Smith           Mary VII      24286.00
64         Brams           Han IV        28054.00
72         Brams           Berg II       25796.00
89         Morzart         John X        31524.00
90         Brams           Tom IV         5023.00
92         Subert          Gord II       31564.00
94         Monteli         Vick II        9281.00
95         Monteli         John II       12007.00
98         Brams           Jane X        32603.00
100        Jhonston        Pey VI        25175.00

Enter your choice
1 - store a formatted text file of accounts
    called "print.txt" for printing
2 - update an account
3 - add a new account
4 - delete an account
5 - end program
? 3 Smar Than 2000
Enter lastname, firstname, balance
? A new account is created Successfully: 80
```



```
Enter your choice
1 - store a formatted text file of accounts
    called "print.txt" for printing
2 - update an account
3 - add a new account
4 - delete an account
5 - end program
? 3 TaChung Kim 4000
Enter lastname, firstname, balance
? A new account is created Successfully: 47

Enter your choice
1 - store a formatted text file of accounts
    called "print.txt" for printing
2 - update an account
3 - add a new account
4 - delete an account
5 - end program
? 2 72 204
Enter account to update (1 - 100): 72       Brams          Berg II      25796.00

Enter charge (+) or payment (-): 72       Brams          Berg II      26000.00

Enter your choice
1 - store a formatted text file of accounts
    called "print.txt" for printing
2 - update an account
3 - add a new account
4 - delete an account
5 - end program
? 4 89
Enter account to delete (1 - 100): Account #89 deleted.

Enter your choice
1 - store a formatted text file of accounts
    called "print.txt" for printing
2 - update an account
3 - add a new account
4 - delete an account
5 - end program
? 3 Hamber Gao 6000
Enter lastname, firstname, balance
? A new account is created Successfully: 84

Enter your choice
1 - store a formatted text file of accounts
    called "print.txt" for printing
2 - update an account
3 - add a new account
4 - delete an account
5 - end program
? 5
```



```
Account    Last Name       First Name     Balance
1          Adams           Berg X         7541.00
2          Adams           Mary IV       26685.00
3          Monteli         Pey II        26111.00
4          Monteli         Han IV         2448.00
5          Monteli         Pey IV        28658.00
6          Brams           Will VI       20950.00
7          Subert          Jane I        11871.00
8          Hamilton        Pey X          4905.00
9          Subert          John VI        9610.00
10         Bach            Berg VI       20759.00
11         Brams           John IX       16234.00
19         Adams           Mary VIII       835.00
20         Heisenberg      John V        18041.00
21         Adams           Mary V        12227.00
29         Jhonston        Pey I         25990.00
30         Subert          Jane VIII      3918.00
31         Bach            Mary II       18843.00
33         Subert          Gord VIII      5320.00
34         Smith           Tom I          8430.00
39         Jhonston        Pey IV        25179.00
43         Smith           Han II        30146.00
47         TaChung         Kim            4000.00
53         Brams           Gord IX       23003.00
56         Monteli         Will I         8807.00
61         Bach            Will X        23280.00
62         Smith           Mary VII      24286.00
64         Brams           Han IV        28054.00
72         Brams           Berg II       26000.00
80         Smar            Than           2000.00
84         Hamber          Gao            6000.00
90         Brams           Tom IV         5023.00
92         Subert          Gord II       31564.00
94         Monteli         Vick II        9281.00
95         Monteli         John II       12007.00
98         Brams           Jane X        32603.00
100        Jhonston        Pey VI        25175.00
```

This account (47) is added.

This account (72) is modified.

These two accounts (80 & 84) are added.

Account 89 is deleted.