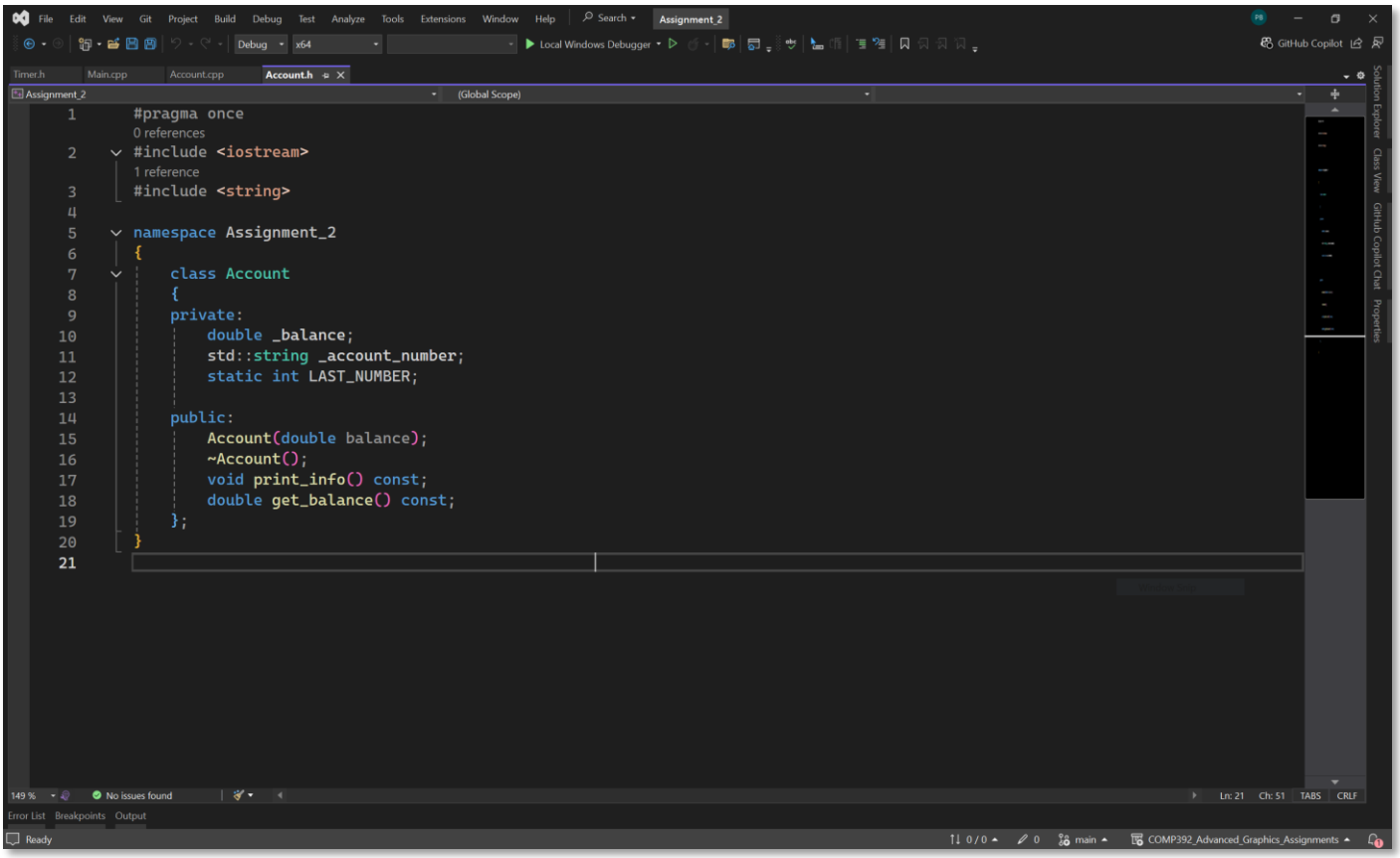
ASSIGNMENT - 2

RAW POINTERS

- Implement an Account class using a C++ program. Match the output against the sample output given the provided test harness.



```
#pragma once
0 references
#include <iostream>
1 reference
#include <string>

namespace Assignment_2
{
    class Account
    {
    private:
        double _balance;
        std::string _account_number;
        static int LAST_NUMBER;

    public:
        Account(double balance);
        ~Account();
        void print_info() const;
        double get_balance() const;
    };
}
```

Fig 1.1: Header file for Account

```cpp
#include "Account.h"

namespace Assignment_2
{
    int Account::LAST_NUMBER = 1000;

    Account::Account(double balance)
    {
        _balance = balance;
        _account_number = std::to_string(_Val: LAST_NUMBER);
        LAST_NUMBER++;

        std::cout << "<<ACCOUNT>> Constructor Invoked for account: " << _account_number << std::endl;
    }

    Account::~Account()
    {
        std::cout << "<<ACCOUNT>> Destructor Invoked for account: " << _account_number << std::endl;
    }

    void Account::print_info() const
    {
        std::cout << "Account: " << _account_number << ", Balance: " << _balance << std::endl;
    }

    double Account::get_balance() const
    {
        return _balance;
    }
}
```

Fig 1.2: CPP file for Account

```cpp
#pragma once
/**
 * Author  : Yan Chernokow
 * Filename: timer.hpp
 * Purpose : provide a simple way to time code
 * Date    : February 2, 2022
 */

#pragma once
#include <iostream>
#include <chrono>

class Timer
{
public:
    Timer()
    {
        m_StartTimepoint = std::chrono::high_resolution_clock::now();
    }

    ~Timer()
    {
        Stop();
    }

    void Stop()
    {
        auto std::chrono::...k::time_point endTimepoint = std::chrono::high_resolution_clock::now();
        auto long long start = std::chrono::time_point_cast<std::chrono::microseconds>(_Time: m_StartTimepoint).time_since_epoch().count();
        auto long long end = std::chrono::time_point_cast<std::chrono::microseconds>(_Time: endTimepoint).time_since_epoch().count();
        auto long long duration = end - start;
        double ms = duration * 0.001;
        std::cout << duration << "us (" << ms << "ms)\n";
    }

private:
    std::chrono::time_point<std::chrono::high_resolution_clock> m_StartTimepoint;
};
```

Fig 1.3: Timer Class for Account

```cpp
#include <iostream>
#include <string>
#include <algorithm>
#include "Account.h"
#include "Timer.h"

void assignmentTask1();
Assignment_2::Account** create_accounts(int size, int max_balance);
void print_accounts(Assignment_2::Account* accounts[], int size);
void sort_accounts(Assignment_2::Account** accounts, int size);
void test_account();
int insertion_sort(Assignment_2::Account* accounts[], int n);

int main()
{
    assignmentTask1();

    return 0;
}

void assignmentTask1()
{
    test_account();
}

void test_account()
{
    srand(301344949);     //use your student number
    const int MAX_BALANCE = 100000;
    const int SIZE = 20;  //change this number to investigate performance
    Assignment_2::Account a1(rand() % MAX_BALANCE);
    Assignment_2::Account a2(rand() % MAX_BALANCE);
    Assignment_2::Account a3(rand() % MAX_BALANCE);
    Assignment_2::Account a4(rand() % MAX_BALANCE);

    a1.print_info();
```

Fig 1.4: Main Class part – 1

```cpp
}

void test_account()
{
    srand(301344949);     //use your student number
    const int MAX_BALANCE = 100000;
    const int SIZE = 20;  //change this number to investigate performance
    Assignment_2::Account a1(rand() % MAX_BALANCE);
    Assignment_2::Account a2(rand() % MAX_BALANCE);
    Assignment_2::Account a3(rand() % MAX_BALANCE);
    Assignment_2::Account a4(rand() % MAX_BALANCE);

    a1.print_info();
    a2.print_info();
    a3.print_info();
    a4.print_info();

    std::cout << "\nCreating the list of accounts:\n";
    Assignment_2::Account** accounts = create_accounts(SIZE, MAX_BALANCE);

    std::cout << "\nPrinting the list of accounts:\n";
    print_accounts(accounts, SIZE);

    std::cout << "\nSorting the list of accounts:\n";
    sort_accounts(accounts, SIZE);

    std::cout << "\nPrinting the list of accounts after sort:\n";
    print_accounts(accounts, SIZE);

    std::cout << "\nDeleting the accounts:\n";
    for (int i = 0; i < SIZE; i++)
    {
        delete(accounts[i]);
    }
}

Assignment_2::Account** create_accounts(int size, int max_balance)
{
    Assignment_2::Account** accounts = new Assignment_2::Account * [size];
```

Fig 1.5: Main Class part – 2

```
Assignment_2::Account** create_accounts(int size, int max_balance)
{
    Assignment_2::Account** accounts = new Assignment_2::Account * [size];

    for (int i = 0; i < size; i++)
    {
        accounts[i] = new Assignment_2::Account(rand() % max_balance);
    }

    return accounts;
}

void print_accounts(Assignment_2::Account* accounts[], int size)
{
    for (int i = 0; i < size; i++)
    {
        accounts[i]->print_info();
    }
}

void sort_accounts(Assignment_2::Account** accounts, int size)
{
    Timer* timer = new Timer();
    int swaps = insertion_sort(accounts, size);
    delete(timer);

    std::cout << "\nNo. of swaps: " << swaps << "\n";
}

int insertion_sort(Assignment_2::Account* accounts[], int size)
{
    int swaps = 0;

    for (int i = 1; i < size; ++i)
    {
        Assignment_2::Account* key = accounts[i];

        int j = i - 1;

        while (j >= 0 && accounts[j]->get_balance() > key->get_balance())
        {
            accounts[j + 1] = accounts[j];
            swaps++;
            j = j - 1;
```

Fig 1.6: Main Class part – 3

```
    return accounts;
}

void print_accounts(Assignment_2::Account* accounts[], int size)
{
    for (int i = 0; i < size; i++)
    {
        accounts[i]->print_info();
    }
}

void sort_accounts(Assignment_2::Account** accounts, int size)
{
    Timer* timer = new Timer();
    int swaps = insertion_sort(accounts, size);
    delete(timer);

    std::cout << "\nNo. of swaps: " << swaps << "\n";
}

int insertion_sort(Assignment_2::Account* accounts[], int size)
{
    int swaps = 0;

    for (int i = 1; i < size; ++i)
    {
        Assignment_2::Account* key = accounts[i];

        int j = i - 1;

        while (j >= 0 && accounts[j]->get_balance() > key->get_balance())
        {
            accounts[j + 1] = accounts[j];
            swaps++;
            j = j - 1;
        }

        accounts[j + 1] = key;
        swaps++;
    }

    return swaps;
}
```

Fig 1.7: Main Class part – 4

```
Microsoft Visual Studio Debug Console
<<ACCOUNT>> Constructor Invoked for account: 1000
<<ACCOUNT>> Constructor Invoked for account: 1001
<<ACCOUNT>> Constructor Invoked for account: 1002
<<ACCOUNT>> Constructor Invoked for account: 1003
Account: 1000, Balance: 10035
Account: 1001, Balance: 20356
Account: 1002, Balance: 21351
Account: 1003, Balance: 29291

Creating the list of accounts:
<<ACCOUNT>> Constructor Invoked for account: 1004
<<ACCOUNT>> Constructor Invoked for account: 1005
<<ACCOUNT>> Constructor Invoked for account: 1006
<<ACCOUNT>> Constructor Invoked for account: 1007
<<ACCOUNT>> Constructor Invoked for account: 1008
<<ACCOUNT>> Constructor Invoked for account: 1009
<<ACCOUNT>> Constructor Invoked for account: 1010
<<ACCOUNT>> Constructor Invoked for account: 1011
<<ACCOUNT>> Constructor Invoked for account: 1012
<<ACCOUNT>> Constructor Invoked for account: 1013
<<ACCOUNT>> Constructor Invoked for account: 1014
<<ACCOUNT>> Constructor Invoked for account: 1015
<<ACCOUNT>> Constructor Invoked for account: 1016
<<ACCOUNT>> Constructor Invoked for account: 1017
<<ACCOUNT>> Constructor Invoked for account: 1018
<<ACCOUNT>> Constructor Invoked for account: 1019
<<ACCOUNT>> Constructor Invoked for account: 1020
<<ACCOUNT>> Constructor Invoked for account: 1021
<<ACCOUNT>> Constructor Invoked for account: 1022
<<ACCOUNT>> Constructor Invoked for account: 1023

Printing the list of accounts:
Account: 1004, Balance: 29129
Account: 1005, Balance: 2335
Account: 1006, Balance: 17087
Account: 1007, Balance: 5928
Account: 1008, Balance: 21890
Account: 1009, Balance: 26980
Account: 1010, Balance: 31705
Account: 1011, Balance: 26973
Account: 1012, Balance: 17564
Account: 1013, Balance: 20527
Account: 1014, Balance: 31604
Account: 1015, Balance: 24027
Account: 1016, Balance: 24433
Account: 1017, Balance: 8188
Account: 1018, Balance: 3945
Account: 1019, Balance: 27434
Account: 1020, Balance: 25152
Account: 1021, Balance: 1183
Account: 1022, Balance: 21133
Account: 1023, Balance: 9685

Sorting the list of accounts:
2us (0.002ms)
```

Fig 1.8: Output of Code part - 1

```
Microsoft Visual Studio Debug Console
Account: 1023, Balance: 9685

Sorting the list of accounts:
2us (0.002ms)

No. of swaps: 123

Printing the list of accounts after sort:
Account: 1021, Balance: 1183
Account: 1005, Balance: 2335
Account: 1018, Balance: 3945
Account: 1007, Balance: 5928
Account: 1017, Balance: 8188
Account: 1023, Balance: 9685
Account: 1006, Balance: 17087
Account: 1012, Balance: 17564
Account: 1013, Balance: 20527
Account: 1022, Balance: 21133
Account: 1008, Balance: 21890
Account: 1015, Balance: 24027
Account: 1016, Balance: 24433
Account: 1020, Balance: 25152
Account: 1011, Balance: 26973
Account: 1009, Balance: 26980
Account: 1019, Balance: 27434
Account: 1004, Balance: 29129
Account: 1014, Balance: 31604
Account: 1010, Balance: 31705

Deleting the accounts:
<<ACCOUNT>> Destructor Invoked for account: 1021
<<ACCOUNT>> Destructor Invoked for account: 1005
<<ACCOUNT>> Destructor Invoked for account: 1018
<<ACCOUNT>> Destructor Invoked for account: 1007
<<ACCOUNT>> Destructor Invoked for account: 1017
<<ACCOUNT>> Destructor Invoked for account: 1023
<<ACCOUNT>> Destructor Invoked for account: 1006
<<ACCOUNT>> Destructor Invoked for account: 1012
<<ACCOUNT>> Destructor Invoked for account: 1013
<<ACCOUNT>> Destructor Invoked for account: 1022
<<ACCOUNT>> Destructor Invoked for account: 1008
<<ACCOUNT>> Destructor Invoked for account: 1015
<<ACCOUNT>> Destructor Invoked for account: 1016
<<ACCOUNT>> Destructor Invoked for account: 1020
<<ACCOUNT>> Destructor Invoked for account: 1011
<<ACCOUNT>> Destructor Invoked for account: 1009
<<ACCOUNT>> Destructor Invoked for account: 1019
<<ACCOUNT>> Destructor Invoked for account: 1004
<<ACCOUNT>> Destructor Invoked for account: 1014
<<ACCOUNT>> Destructor Invoked for account: 1010
<<ACCOUNT>> Destructor Invoked for account: 1003
<<ACCOUNT>> Destructor Invoked for account: 1002
<<ACCOUNT>> Destructor Invoked for account: 1001
<<ACCOUNT>> Destructor Invoked for account: 1000

E:\College\Semester_6\COMP392_002_Advanced_Graphics\COMP392_Advanced_Graphics_Assignments\Assignment_2\x64\Debug\Assignment_2.exe (process 38676) exited with code 0 (0x0).
```

Fig 1.9: Output of Code part - 2