Data Structures – Spring 2023
Project 3
Due: March 10th 2023 at 11:59 PM

Objective:
The goal of this project is to create a single BlockChain using the given class structures with linked lists. An extension of this project will be project 4.

Description:
Blockchain is a revolutionary technology that has gained widespread attention in recent years. It is a decentralized digital ledger that enables secure and transparent transactions without the need for intermediaries such as banks or governments. The underlying technology is based on a distributed network that is replicated across a network of computers/nodes. This makes blockchain ideal for applications that require transparency, security, and trust in a decentralized environment, such as cryptocurrency, supply chain management, and voting systems. The potential uses of blockchain are vast and have the potential to revolutionize various industries by creating a more secure, efficient, and transparent way of doing business.

In this project, we have three classes that you can see in the boilerplate code. The **transaction class** will hold information about one transaction. An array of transaction class objects will be present in the **block class**. This block class has a block number, the current number of transactions and the maximum number of transactions that the block can hold. This maximum number of transactions per block is given in the input.
The **blockchain class** has a list (linked list) of blocks along with the number of blocks. Object of this blockchain class is what we declare in the main function.

For this project, you need not maintain the value of each person ID (fromID and toID). You will just read the transaction from the input and insert into the data structure (blockchain). You need not worry about if the person ID has enough funds to make the transaction or not.

Input explanation:
The input file given is in the following format:

```
10              <— number of transactions per block
15              <— total number of transactions

10001 111 222 50 0042:01012021
10002 113 111 80 0142:01012021
10003 114 111 20 0245:01022021
10004 222 111 10 0143:01022021
10005 222 113 10 0144:01022021
10006 212 113 70 0244:01022021
10007 212 113 40 0245:01022021
10008 212 113 70 0334:01022021
```

```
10009 212 113 70 0445:01032021
10010 212 113 70 0555:01032021
10011 111 222 50 0142:01032021
10012 113 111 80 0242:01042021
10013 114 111 20 0345:01042021
10014 222 111 10 1143:01042021
10015 222 113 10 2144:01042021
```

The input file starts with the number of transactions per block and then the total number of transactions given in the input file. This is followed by the list of transactions.

Each transaction has the transaction ID, followed by the fromID, toID, amount to be transferred, and finally the time stamp. All these are integers except the last one that is a string and will stay as a string throughout.

To get you all started, a **boiler plate file** is given as well. This file is also given alongside this project. Keep up with the FAQ discussion for the project as more assumptions will be updated there. Sample output file will be released soon as well.

Redirected input:

Redirected input provides you a way to send a file to the standard input of a program without typing it using the keyboard. To use redirected input in Visual Studio environment (on windows), follow these steps: After you have opened or created a new project, on the menu go to project, project properties, expand configuration properties until you see Debugging, on the right you will see a set of options, and in the command arguments type <"input filename". The < sign is for redirected input and the input filename is the name of the input file (including the path if not in the working directory).

If you use macOS or linux (or windows using powershell), you may use the terminal to compile and run your program using g++. This is the compiler that we use in the autograder on GradeScope (you may need to install g++ on your computer). Once you installed g++, you may compile your program using the following command:

g++ project3.cpp -o p3

You may then run the program with the redirected input using the following command:

./p3 < input1.txt

For powershell,

Get-Content input1.txt | ./p3

Make sure your program and your input file are in the same folder.

You may also use Google CoLab as an alternative and a sample file is given along with the project. This is an ipynb file which has to be opened in Google CoLab.

A simple program that reads the input file (one number) and displays everything that was read is given below.

```
#include<iostream>
using namespace std;

int main()
{
  int num = 0;
  cout << "Hello World" << endl;
  cin >> num;
  cout << num;
  return 0;
}
```

Sample output file for corresponding input files will be released. The input1.txt file given is a simple input file that will help you understand the project, more complicated ones will be released later and used for grading.

Submission:

Submission will be through GradeScope. Your program will be autograded with test cases and then hand graded to check for documentation and if you have followed the specifications given. You may submit how many ever times to check if your program passes the test cases provided. Test cases will be released at the beginning and later other test cases will be released while grading. For the autograder to work, the program you upload must be named as **project3.cpp**. This will be the only file you will submit. Make sure it is well commented.

Constraints:

1. In this project, the only header you will use the header files that are given to you in the boiler plate code. Using other or excess header files will be subject to a heavy grade penalty for the project.
2. None of the projects is a group project. Consulting with other members of this class on programming projects is strictly not allowed and plagiarism charges will be imposed on students who do not follow this.
    - Please review academic integrity policies in the modules.

How to ask for help:

1. Check FAQ discussion board of the project first.
2. Email the TAs with your precise questions.
3. Upload your well commented and clean program to CodePost.
    a. This is a website which is used to share code with your TA only.
    b. You will upload your program and I can view it and comment on it.
    c. Here is the invite link to our class for the summer.
    d. Once you join the class on CodePost, you can upload your program to the Project 1 assignment.

        Note: Your program will **not** be auto graded at CodePost, this is just for when you want to ask a question and a place where I can look at your program and comment on it. CodePost is great for live feedback.

GradeScope is the place where you should submit and where your program will be autograded.