CS 2413 – Data Structures – Fall 2024 – Project Two
Due 11:59 PM, September 27, 2024

Description: Imagine you have a collection of specialized computing chips designed to perform a specific elementary arithmetic operation. These operations include addition, subtraction, multiplication, negation, and division. Each chip, except for the negation chip, requires two input values and produces a single output value. Due to the nature of its operation, the negation chip only needs one input and generates one output.

Among these chips, there is a unique type called the output chip. This chip has one input and no output, as its sole function is to display the value it receives through its input line. For instance, if an output chip labeled O50 receives a value of 987.2, it will display a message like: "I am output chip number 50, and the value I received is 987.2."

In addition to the output chip, an input chip is designed to provide values to the computing system. This input chip feeds a value into the circuit, acting as a data source for the calculations performed by the other chips.

Consider a configuration where four chips are connected in sequence to perform a series of operations. These chips are as follows:

1. **A100 (Addition Chip):** This chip adds two input values. It is connected to two input chips labeled I1 and I2. These input chips supply the values to be added.
2. **N110 (Negation Chip):** The output from the addition chip A100 is sent to the input of the negation chip N110. The negation chip takes this input value and negates it, producing a single output that is the negative of the input value.
3. **M120 (Multiplication Chip):** This chip requires two inputs to perform multiplication. One of its inputs is connected to the output of the negation chip N110, while the other input comes from the output of another input chip labeled I3.
4. **S130 (Subtraction Chip):** The output from the multiplication chip M120 is fed into one of the inputs of the subtraction chip S130. The other input to this chip comes from the output of an additional input chip labeled I4. The subtraction chip then subtracts the second input from the first and produces an output.

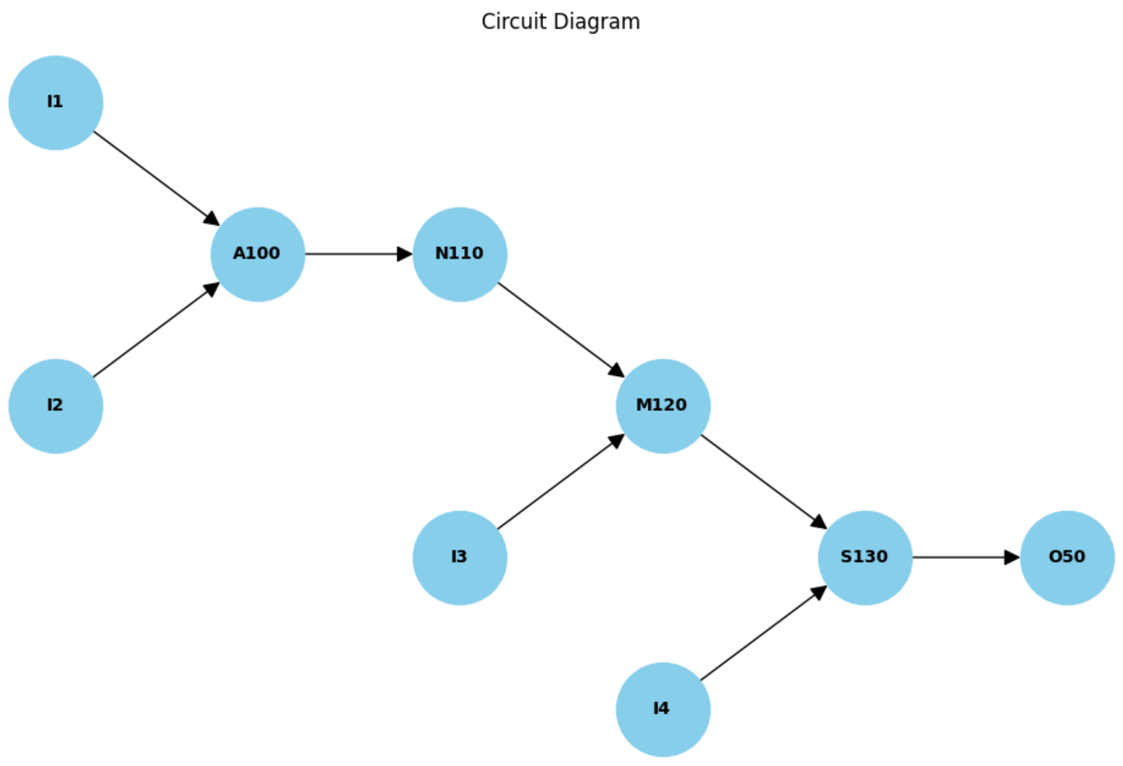
Finally, the output from the subtraction chip S130 is connected to the input of the output chip, labeled O50. This output chip, O50, receives and prints the final computed value from the sequence of operations. For instance, if the final value is 42.0, it would display: "I am output chip number 50, and the value I received is 42.0."

In summary, the configuration of these chips allows a series of arithmetic operations to be performed in sequence, where each chip's output is passed as an input to the next chip in the chain. The entire process culminates in the output chip, which displays the final result of the computations.

The input to your program will have the following format for the above. First, we build the circuit. Note that the first character is to add a chip to our circuit with a single letter A.

```
9         <- The number of Chips
I1
I2
I3
I4
A100
M120
N110
S130
O50
13              <- Number of commands including input command and the output command
A I1 A100
A I2 A100
A A100 N110
A N110 M120
A I3 M120
A M120 S130
A I4 S130
A S130 O50
```

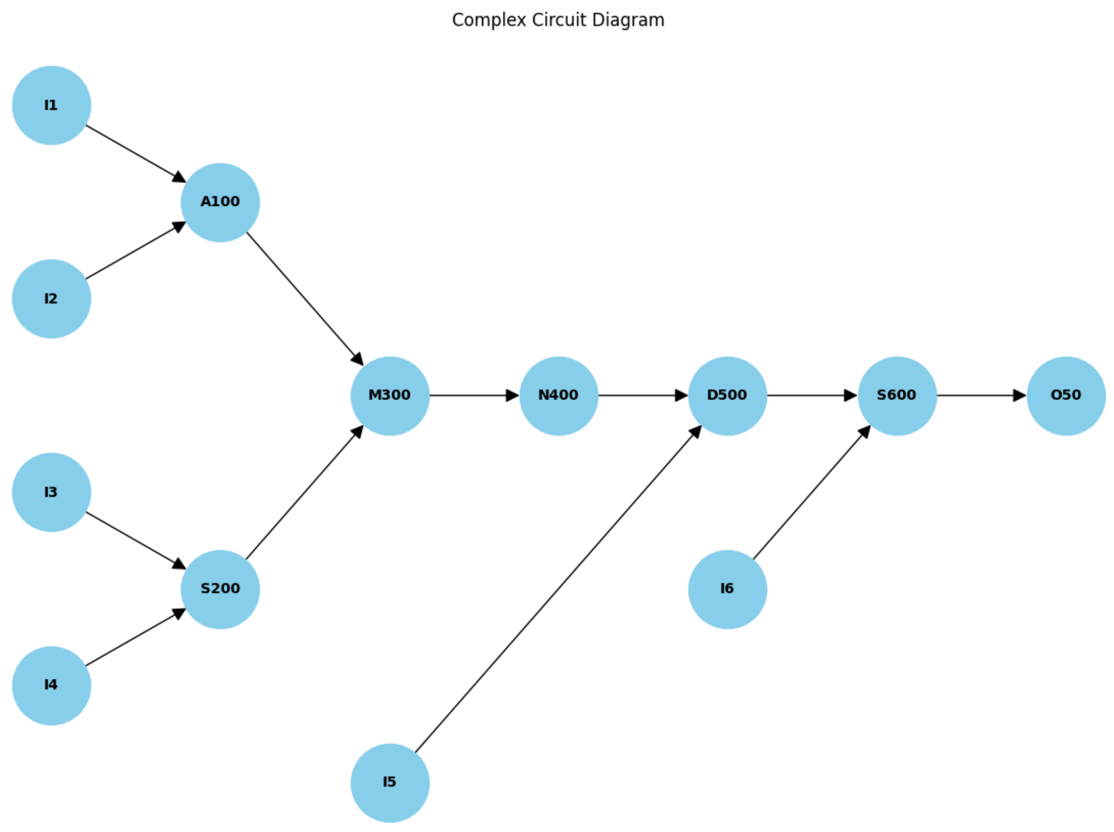
The above circuit is pictorially shown below.



After you build the circuit, you give input values, let the circuit execute, and produce an answer. The input is shown below. Note that the first character is an I, which tells us we are feeding some values into the input chips. You will also see that the first character is an O, which tells us we are asking the output chip to give us the output.

I I1 5
I I2 10
I I3 5
I I4 10
O O50

Now consider a much more elaborate circuit, as shown pictorially below.



The input for building the above circuit will be as follows:

13. <- Number of chips in our circuit
I1
I2
I3
I4
I5
I6
O50
A100

```
S200
M300
N400
D500
S600
19            <- Number of commands including input command and the output command
A I1 A100
A I2 A100
A I3 S200
A I4 S200
A A100 M300
A S200 M300
A M300 N400
A N400 D500
A I5 D500
A D500 S600
A I6 S600
A S600 O50
```

Once you provide input values for I1, I2, I3, I4, I5, and I6, as below, you should produce the correct output.

```
I I1 5
I I2 10
I I3 3
I I4 1
I I5 4
I I6 2
O O50
```

Your Project Implementation: As part of this project, you will create the CHIP class as described below.

Programming Objectives:

1. All code must be in C++.You will create a class given below with appropriate fields.
2. The class will have several methods whose prototypes are provided to you.
3. All input will be read via redirected input. That is, you will not open a file inside the program.
4. The class structure is shown below (you are responsible for fixing any syntax errors).
5. The structure for your main program is also provided. You will use that for your project.

Program Structure:

You will have the following class structure. You have to ensure that all methods and appropriate additional fields are added.

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

class Chip {
private:
    char chipType;    // Type of the chip (A: Addition, S: Subtraction, etc.)
    string id;        // Unique identifier for the chip
    Chip* input1;     // Pointer to the first input chip
    Chip* input2;     // Pointer to the second input chip (can be NULL)
    Chip* output;     // Ptr to the output chip (is NULL for output chips)
    double inputValue; //for the input chip

public:
    // Constructor
    Chip(char type, const string& id);

    // Method prototypes
    void setInput1(Chip* inputChip);  // Sets the first input chip
    void setInput2(Chip* inputChip);  // second input chip (can be NULL)
    void setOutput(Chip* outputChip); // Sets the output chip (can be NULL)
    void compute();    // Performs the operation based on the chip type

    void display() const; // Displays the chip's information
      //example: I6, Output = S600 --- for the input Chip
      //example: O50, Input 1 = S600 --- for the output Chip
      //example: A100, Input 1 = I1, Input 2 = I2, Output = M300
    string getId() const;    // Returns the chip ID

//****** OTHER METHODS AS NECESSARY ************//

};
```

The compute( ) method is the key to your program. It is first executed on the output Chip, that is, the object corresponding to CHIP, say O50.

Your main program will have the following structure: You will need to write plenty of code and make changes.

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

//Define your Chip class and all its methods.

int main () {

      //**** ALL THE VARIABLES YOU NEED FOR THIS MAIN FUNCTION *****//

      int numChips;
      Chip** allChips;
      int numCommands;

      cin >> numChips;
      //create an array Chip objects pointers
      allChips = new Chip*[numChips];
      //each array location is a pointer to a Chip Object

      for (int i=0; i < numChips; i++) {
         //read the chip ID based on the first letter to determine its type
         //create the chip object and initialize it appropriately
         //store the chip object in the allChips array
      }

      for (int i=0; i < numCommands; i++) {

            // read from input the links and make the appropriate
            //connections. You may need to search the array allChips
            //to find the chip that is referred and connect.

            // If the first letter is an O, then execute the compute method
            // on the object referred.
      }

      cout << "***** Showing the connections that were established" << endl;
      //for each component created call the display () method

      return 0;
}
```

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, 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).

Constraints

1. In this project, the only header you will use is #include <iostream> and #include <string> and using namespace std.
2. None of the projects is a group project. Consulting with other members of this class our seeking coding solutions from other sources including the web on programming projects is strictly not allowed and plagiarism charges will be imposed on students who do not follow this.

Project Submission Requirements:

1. **Code Development (75%):** Implement the provided class structure for the Chip by writing the necessary methods to manipulate the Chip circuit. Your implementation must be fully compatible with the main program provided, which is designed to create and manipulate Chip objects. Your code must run successfully with the main program and the corresponding input, demonstrating the correct functionality of the Chip class and its methods.

- **LLM/AI Tool Usage:** You can use Large Language Models (LLMs) or AI tools, such as GitHub Copilot, to assist in writing and refining your classes and their methods. If you did not use LLM or AI tools to write your project, you still have to show for 2. below how you would have used it to find a solution to the project. **You need to make sure that you use the class structure provided to you as the basis, and failure to do that will result in zero points for this project.**

2. **LLM and GitHub Copilot Usage Documentation (15%):** If you choose to use LLM tools or GitHub Copilot, you must document your usage. This documentation (in PDF Format) should include:

   - **Prompts and Suggestions:** Provide the specific prompts or suggestions you used, such as "Generate a method for this method" or "How can I implement a this function for a Chip class?"
   - **Rationale:** Explain why you chose these prompts or suggestions and how they contributed to the development of your classes. For instance, you might describe how a particular suggestion helped you structure the program.
   - **Incremental Development:** Detail how you used the tools to build and refine your classes and methods incrementally. For example, you might start by generating a basic structure for the classes and then refine individual methods, ensuring each one integrates smoothly with the main program.

3. **Debugging and Testing Plan (10%):** Submit a comprehensive debugging and testing plan. This should include:

   - **Specific Tests:** Describe the tests you conducted on your class methods, such as checking that the compute method works correctly for various circuits.
   - **Issues and Resolutions:** Document any issues you encountered, such as handling zero values or optimizing for performance, and how you resolved them.
   - **Verification:** Explain how you verified that your classes work correctly with the provided main program. This could involve running a series of test cases the main program provides or creating additional test cases to ensure robustness.