PROGRAM 4: Jugs

In the movie “Die Hard 3”, Bruce Willis and Samuel L. Jackson were confronted with the following puzzle. They were given a 3-gallon jug and a 5-gallon jug and were asked to fill the 5-gallon jug with exactly 4 gallons. The solution they reached was to fill the 5-gallon jug, pour the 5-gallon jug into the 3-gallon jug (this left 2 gallons in the 5-gallon jug), empty the 3-gallon jug, pour the 5-gallon jug into the 3-gallon jug, fill the 5-gallon jug and then fill the 3-gallon jug from the 5-gallon jug, empty the 3 gallon jug, leaving 4 gallons in the 5 gallon jug.

Your assignment will be to generalize this problem in the follow manner.

You have two jugs, A and B, and an infinite supply of water. There are three types of actions that you can use, each with a cost:

1. you can fill jug A or B
2. you can empty jug A or B
3. you can pour from one jug to the other.

Pouring from one jug to the other stops when the first jug is empty or the second jug is full, whichever comes first. For example, if A has 5 gallons and B has 6 gallons and a capacity of 8, then pouring from A to B leaves B full and 3 gallons in A.

A problem is given by (Ca, Cb, N, cfA, cfB, ceA, ceB, cpAB, cpBA), where Ca and Cb are the capacities of the jugs A and B, respectively, and N is the goal. cfA is the cost to fill A, cfB is the cost to fill B, ceA, is the cost to empty A, ceB is the cost to empty B, cpAB is the cost to pour A to B and cpBA is the cost to pour B to A. A solution is a sequence of steps that leaves jug A empty, and exactly N gallons in jug B. The possible steps are

- fill A
- fill B
- empty A
- empty B
- pour A B

- pour B A
- success X

fill means to fill the jug from the infinite water supply

empty means to discard the water in the jug

"pour A B" means "pour the contents of jug A into jug B"

"success X" means that the goal has been accomplished, jug B contains N gallons at a cost of X.

Program Specs

You **MUST** solve this problem by building and traversing a graph, if you find a cute math formula to solve the problem **NO** credit will be given.

Write a Jug class that takes 9 parameters in the constructor. (Ca, Cb, N, cfA, cfB, ceA, ceB, cpAB, cpBA) Note: the order **IS** important. Also checking for valid input is something you should do.

Your Jug class must have a solve() member which will solve and then store the steps for the CHEAPEST solution in a string.

```
class Jug {
    public:
        Jug(int,int,int,int,int,int,int,int,int);
        ~Jug();

        //solve is used to check input and find the solution if one exists
        //returns -1 if invalid inputs. solution set to empty string.
        //returns 0 if inputs are valid but a solution does not exist. solution set to empty string.
        //returns 1 if solution is found and stores solution steps in solution string.
        int solve(string &solution);
    private:
        //anything else you need
};
```

You may also find that you need an few helping structures to aid in the building of your graph. You may use queues, stacks, heaps, priority queues, trees or any other **NON GRAPH** structure you need. You make take these from old homework, or the STL. Also you **MUST** state in your comments at the top of the Jug.h file that you are using them and give credit where credit is due.

Input

Input to your program consists of 9 ints defining one puzzle. Ca, Cb, N, cfA, cfB, ceA, ceB, cpAB, cpBA. Ca and Cb are the capacities of jugs A and B, and N is the goal. the rest are the costs for each action. You will want to verify the costs are positive and $0 < Ca \le Cb$ and $N \le Cb \le 1000$. If the inputs are invalid solve will return -1.

Solution

If a solution exists, the solve function should generate a string, that if output would look like the sample output below. This string will store the steps as a series of instructions, one instruction per line, from the list of the potential output lines which will result in jug A being empty and jug B containing exactly N gallons of water. The last line for each puzzle should be the line "success X". (where X is the cost). There should be no empty lines nor any trailing spaces within this string. This string is "returned" via the reference parameter.

main.cpp

```
{
   string solution;
   Jug head(3, 5, 4, 1, 2, 3, 4, 5, 6);
   if (head.solve(solution) != 1) {
      cout << "Error 3" << endl;
   }
   cout << solution << endl << endl;
}
{
   string solution;
   Jug head( 3, 5, 4, 1, 1, 1, 1, 1, 2);
   if(head.solve(solution) != 1) {
      cout << "Error 3" << endl;
   }
   cout << solution << endl;
}
```

Sample Output

If the string generated by the solve function is output, the output would look like this for the first set of values in the above main function:

```
fill A
pour A B
fill A
pour A B
empty B
pour A B
fill A
pour A B
success 27
```

Sample Output Not Shortest Path

```
fill B
pour B A
empty A
pour B A
fill B
pour B A
empty A
success 28
```