ITU Computer Engineering Department
BLG 223E Data Structures, Fall 2018
Homework #4
Due December 13, 2018 11:59pm

Problem Defintion

N ants numbered $1, 2, 3, \ldots, N$ are trying to cross a road that contains several holes with different depths. To pass through a hole with depth $M < N$[1], the M leading ants have to go into the hole so that the remaining ants can safely cross the hole. Then, **the last ant** who crossed the hole (i.e., Nth ant in the line) pulls the ant which is on top (i.e., Mth ant in the line) from the hole. Afterwards, the rescued ant (i.e., Mth ant) pulls the ant on the top (i.e., $(M-1)$th ant) from the hole, and so on until no ants are left in the hole. Then they continue their trip to the next hole[2], and apply the same procedure there. For example, look at the scenario below where $N = 5$ ants are crossing two holes with depths $M = 2$ and $M = 1$.

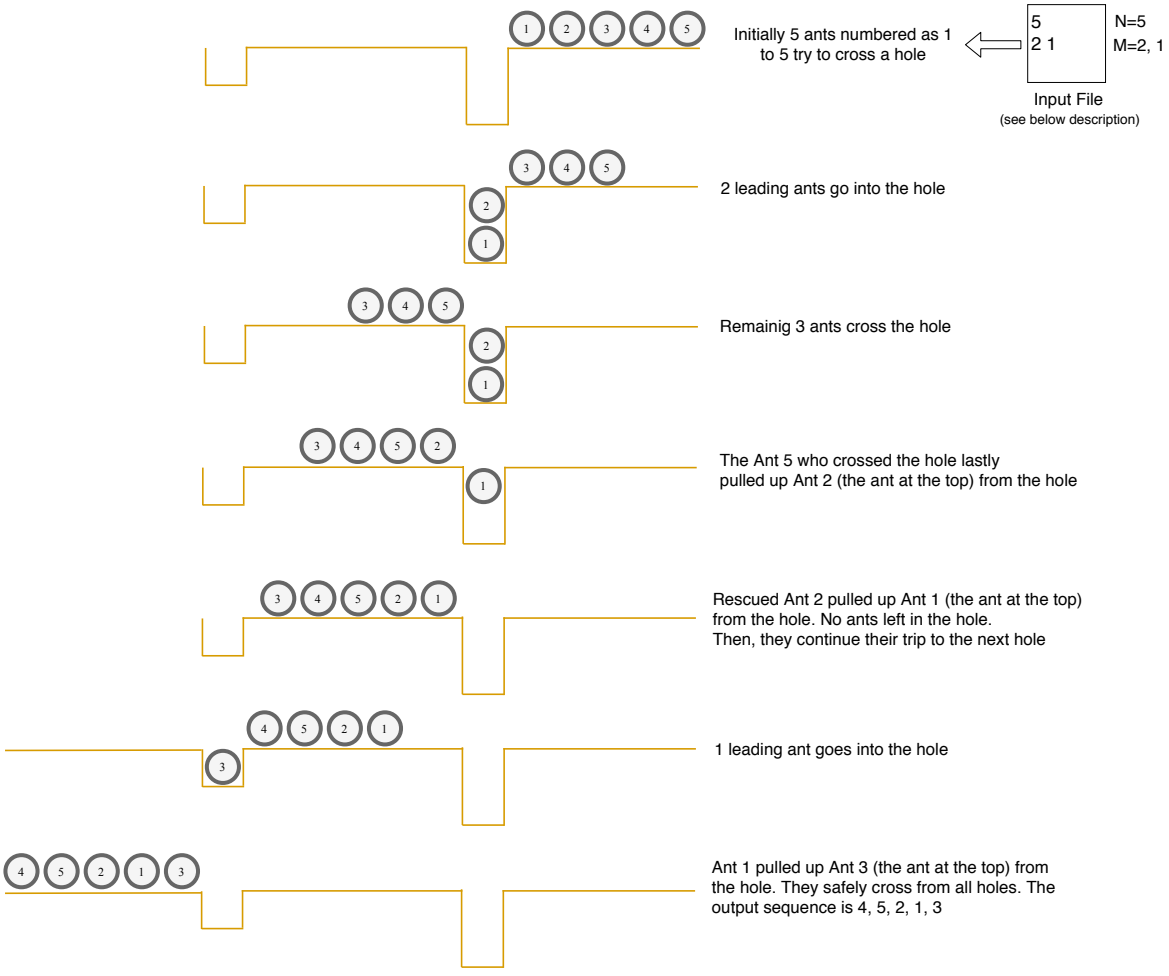

Figure 1: $N = 5$ ants are crossing two holes with depths $M = 2$ and $M = 1$

[1]The depth of the holes cannot be larger than the total number of ants N
[2]The distance between two holes can not be smaller than the number of ants in the ant sequence.

Implementation

1. Use the below data structure `Ants`. Implement `queueAnt` and `stackAnt` data structures. You are going to implement queue and stack data structures respectively. Queue and Stack implementations will be done with singly linked list implementations.

   You are not allowed to include any Standard Template Library (**STL) container**.

```
struct Ants{
        queueAnt ants;
        queueAnt holeDepths;
        stackAnt hole;
        void ReadFile(char*);
        void ShowContents(bool);
        void CrossRoad();
};
```

2. Implement `void ReadFile(char *)`:

   Read number of ants and the depths of holes **dynamically** from the input file whose name is passed as a command line parameter. The first line indicates the number of ants while the second line indicates the depths of holes on the way separated by space character. You can use `C++` classes to perform output and input of characters to/from files.

3. Implement `void ShowContents(bool)`:

   Prints out the contents of the queues according to `bool` argument. If true, prints out the contents of queueAnt `ants`; otherwise prints out the contents of queueAnt `holeDepths`.

4. Implement: `void CrossRoad()`:

   This function makes all ants cross the road that contains several holes whose depths are obtained from the second line of the input file.

Main Program, Compilation and Screen Output

- You are going to use the below main program to test your implementations. Your source code file must be a single file and has the name `hw4.cpp`.

```
int main(int argc, char**argv){
        Ants a;
        a.ReadFile(argv[1]);//store the number of ants and depths of holes
        cout << "The initial Ant sequence is: ";
        a.ShowContents(1); //list ant sequence (initially: 1, 2, ..., N)
        cout << "The depth of holes are: ";
        a.ShowContents(0); //list depth of holes
        a.CrossRoad();
        cout << "The final Ant sequence is: ";
        a.ShowContents(1);

        return 0;
}
```

- Your program should compile and run on Linux environment using **g++ (version 4.8.5 or later)**. You can test your program on ITUs Linux Server using SSH protocol. To compile the code, use the following command:

```
g++ -Wall -Werror hw4.cpp -o hw4
```

- **Your input data filename will be supplied from the command line. Don't name it in your program statically**. Your program must take input filename as command line argument. Then run your executable (named as hw4) using the following command:

  ```
  ./hw4 input_file_name
  ```

  where **input_file_name** is name of any test file we will use to evaluate your homeworks. **IF YOUR PROGRAMS DON'T TAKE FILE NAME FROM THE COMMAND LINE, THEY WILL NOT BE EVALUATED AND BE GRADED AS 0.**

- Main program are going to print out the following message for the given scenario above:

```
The initial Ant sequence is: 1 2 3 4 5
The depth of holes are: 2 1
The final Ant sequence is: 4 5 2 1 3
```

Your program will be checked using **Calico(https://bitbucket.org/uyar/calico)** automatic checker. Therefore, make sure you **print the messages exactly as given in the homework definition** You get zero marks if the automatic checker fails.

Submission Rules

- If a question is not clear, you can ask your question under the thread that is specially started for homework 4 (Questions about HW4) on the message board for BLG 223E on NINOVA. Please check before writing your question whether your question is asked by someone else.

  Do not share any code or text that can be submitted as a part of an assignment (discussing ideas is okay).

- Make sure you write your name and number in all of the files of your project, in the following format:

  /* @Author
  Student Name: <student_name>
  Student ID : <student_id>
  Date: <date> */

- Only electronic submissions through Ninova will be accepted no later than December, 13 at 11:59pm.
- You may discuss the problems at an abstract level with your classmates, but you should not **share or copy code** from your classmates or from the Internet. You should submit your **own, individual** homework.
- Academic dishonesty, including cheating, plagiarism, and direct copying, is unacceptable.
- Use comments wherever necessary in your code to explain what you did.
- Note that **YOUR CODES WILL BE CHECKED WITH THE PLAGIARISM TOOLS!**