CMPE 300
Project 2

Nurlan Dadashov	2019400300
Aziza Mankenova	2018400387

(a) Introduction:

We have completed the project. The code is running successfully.

Run the following command :

mpiexec -n [P+1] python two_towers.py [input.txt] [output.txt]

(b) Structure of the Implementation:

We have chosen the “**checkered**” approach to split processors.

We have a manager and worker processes. The job of the manager process includes reading input file, sending separate blocks of information to worker processes, receiving the final result from the worker processes, and printing the result to output.txt. The worker processes have to receive the data from the manager, carry out the simulation of the game while communicating with the neighboring processes and keeping track of all the changes and send the results to the manager process at the end of the simulation of all the waves.

The manager process is the one with rank = 0. The manager process reads the input file and obtains the size of the map (N), number of waves (W), the number of towers(T) and the coordinates of all the "+"s and "o"s. The manager sends the obtained information about the size of map, number of waves and the size of blocks to worker processes. Next, a three-dimensional array of size (W,N,N) which stores the content, health, and attack points for each cell in all waves is constructed. Then, after splitting the grid into (N/x, N/x) blocks, the worker_data array which stores the content of blocks in each wave for each processor is obtained. This way the information of the content of the block the processor is responsible for is sent to each processor.

Each processor then receives the data from the manager process. The data is a three-dimensional array of size (W, size of block, size of block). The position of the neighbor process is in the given direction is stored in a dictionary. The simulation is run for 8 rounds in each wave. In the beginning, there is an exchange of neighboring data with other workers as a result of invoking the "communicate" function. Next, the damage each tower takes is calculated for each cell. In order to correctly obtain the damage, firstly the id of the cell is calculated. Id represents the position of the cell on the map ($\sqrt{n}$ * row + column). Then, based on the id of the cell and the number of blocks on one side of the map, "get_neighbors" function returns the positions of neighbors. "get_neighbors" was also used to get neighboring blocks and can be reused here as the logic is the same. Next, there is an iteration over all the positions of neighbors of each cell. After calling the "get_opponent" function, the information about the opponent at that specific neighboring cell is obtained and the damage is calculated taking into account the content, position, and attack points. And if health became 0, the tower is removed. Then, the process is repeated for all the waves. In the end, the result of the last wave is sent to the manager process. After receiving the final result from worker processes, the manager process writes the final map into a file.

One of the functions used was "get_neighbors" function. This function returns the positions of all neighboring cells. So, it says if the given call has a neighbor on the left, right, top, bottom, bottom-left, bottom-right, top-left, and/or top-right of it. To obtain the list of neighboring positions, the id of the cell and the number of cells on one side of the map should be passed as parameters.

Another useful function that was utilized is “communicate” function. The most important part of the project is performed in this part. “Communicate” requires one argument which is the wave number, which we use to get data for specific wave and exchange it among the processes.

For example, let’s consider this map:

The processes per each row are as follows:

Row 0: 1, 2, 3, 4	Column 0: 1, 5, 9, 13
Row 1: 5, 6, 7, 8	Column 1: 2, 6, 10, 14
Row 2: 9, 10, 11, 12	Column 2: 3, 7, 11, 15
Row 3: 12, 14, 15, 16	Column 3: 4, 7, 12, 16

Then the even rows are rows 0 and 2, while the odd rows are rows 1 and 3. And, the even columns are columns 0 and 2, while the odd ones are columns 1 and 3.

In the beginning, even rows exchange the data with bottom-odd rows. Firstly, even rows send while bottom odd rows receive. Then, these odd rows send while top even rows receive.

Then, the even columns exchange the data with the right odd columns. Firstly, even columns send while right odd columns receive. Then, these odd columns send while left even columns receive.

Subsequently, even rows exchange data with top odd rows except for the first row. Firstly, even rows send(except the first one), while top odd ones receive. Then, odd rows(except the last row) send to the bottom-even rows, while the even ones receive.

Lastly, even columns exchange data with left-odd columns except for the first column. Firstly, even columns(except the first column) send while the left odd ones receive. Then, odd columns(except the last column) send while the right even ones receive.

This way, the “communicate” function ensures the exchange of all required neighbor data without any deadlocks.

Another useful function is “get_opponent” function, where the opponent information is obtained. Four arguments should be given, data of the process, neighbor’s position, and the coordinate of the cell. The opponent information is either contained in the process’ data or in the neighbor data which was obtained during the communication with other processes.

(c) Analysis of the Implementation:

Communication between the processes contributes most to the time complexity. Time complexity for our communication scheme is O($\sqrt{n}$). There are $\sqrt{n}$ cells in one row or in one column. We are sending data either from one row to another or from one column to another as stated in the previous section. So, we are performing $\sqrt{n}$ operations 8 times (even row $\rightleftarrows$ odd bottom row; even column $\rightleftarrows$ odd right column; even row $\rightleftarrows$ odd top row; even column $\rightleftarrows$ odd left column).

Communicate function is called at each round of each wave (W*8 times). Thus, overall complexity is O(W*$\sqrt{n}$).

For large N values, the program is slow due to the communication overhead. However, dividing work among many processes speeds up the execution.

(d) Test Outputs:

Output 1:

```
+ + . o
+ . . o
. . o .
. o o o
```

Output 2:

```
o . . . . . . o
. . . . . o o .
. + . . . . . .
+ + . . . . . .
+ + + + + + . o
+ . . . . . . o
. . o o . o o o
o o o o o o . o
```

Output 3:

```
+ + + + + + + + . + + + + + + +
+ + + + . . . . + . + + + + + +
+ + + + + + . + . + + + + + + .
+ + + + . + + + + + + . + . + +
. . . + + + + . + + + + . + . +
o . + + + + . + + + + + + . + +
. . . + + + + + + + + + . . . +
. . . + + + + + + . . + . o . +
+ + + + + + + + + + . . . . . +
. . . . + . . . . . . . . + + +
. o . . . + . o o o . . + + . +
. . . . . . . . . . . . + + + +
+ . + . + . + . . + . + + + . +
+ . . + + . . . . . + + + + . +
+ . + + . . + . . . + + + + + +
+ + + + + . . . . . + + . + . +
```

Output 4:

```
o o o . o . . . + + . + . . . o . . . . o . . . . o . . . + . + . . o o . + . . o o o o o . + + + + . . . . . . . + . o . o . .
. . o o . o . . + + + + + . . . o . . o . . . o . . . . + + + + . . o o . . . o o . . o . . . . . . . o o . + + . + . o . . o o
. . . o . . . . . + + . + . . . o o o . . . . . . . o . . . + + . o o o . . o . . o o . o . . . + . o o . . . . . + . o o . . .
. . . o . . + . . + . + . . + . o . . . . o . . o o o . . + . . . o . o . . . o . o . . o . o . . . . . . + . . . . . . . . . .
. . . . . . . . . . . . + + + . . o . o . . . . o o . o . + . . . . o . . . . . . . . . o . o . + . . . . + . . . . . . + . . +
+ . + + . . . . . . + . + . . . . . . . . + . . . . o . . . . o . . o . . . . . + + . . o o . . . . o . + + . o . o o . . . . .
. . . . . + . . o . . + + + . + . . . . . . . o o o . . . . o . . . . . o o . . . + . o . . . . + . . . . . . . . . o . . . + +
+ . o . . . . . o o . . + + + . + . o . o . . . . o . o . . . . . + . o . . . . . . . . . o o . . . . . o . . + . o . . . . + +
+ . . . o . + . . . . + . + + . + . . o o . . . o . . . . . . . . . . . . + + + + + . . . . . . o o . . . . . . . o o . . . + +
+ . . . o . . . . . . . . + + + . . . . . . . . o o o . o . . o o . . . . + + + + . . + . . . . . . . . . + . . . o o . . . . .
+ + . o o . . . . . o o . + + + + + . + + . . . . . o . o . . o . . o o . . . + + + . . . . . . o . . + . . . . o . . o . + + .
+ . . . . . . . o o o o . + . + . . . . . + + . . . o o o o o o . . . . . + . + + + . o o o . o o o . . . . . o o o . . . + + +
. . o . . . o . o o o o . . + . . . . . . . . . o . o o o o o . . . + . . . + + . . . . . . o o . o o o o o o o . . o . . + + +
o o . . . o . o . . . o o . . . . + . . . o . . . . . . o . . . . . . . . . . . . . o . . . . . . o o o o o o . . . o . + + + +
. . . . o o . . o . o . o . . . . . . o o . . . . . . . . . . . . . o . . . . o . . . . . . . o . o o o o o . . . . . . . + . +
. + + . o o o o . . o . o . . . . o o o . . . + . + + . o . . . . . . o o o o . . . . . + . o o . o o . . . o . . . o o . . + +
+ + + . . o o o . . . o o o . + . . . o o . . . . + . . . . o . . . . . o . . . o . o . . . . o o o o o . . . o . . . . o . + +
. . + . . . . . . . . o . o . . . . . o o o . . . . . o . . . . + . o o o . + . o . . . . . . . . o . o o . . o . + + . . . . +
. . . . . + . + . o o o o . . . . . o . o . o o . . o o . . o . . . . o . . . . o o . + . + . + . o . . . o . o . . . . o . + +
. o . . + . . . . . . o o . . . o o o . . o o o o o . . . o o . . o . . . . o . . . . . + . . . . . . . . . . o . + . . o . + +
. . . . . . . . . o o o . . . . . . o . . . o o o o . o o . . . o o . o o o . . . . + + . . . + . + + . . . . . . . + . . . . +
. . . o o o . . . . o o . . . . . . o o . . o o o . o . . . . . o . . o . . . . . . . + . o . . . . + + + . o o . . + + . . + +
o . o o o o o o . o o . . o . . . o o o o o o o o o o o o . . o . . . . . + . . . . . . . . o o . . . . . . o . . . . + + . + +
. . . o o o . . . . . . . . . . o o o o o . o . . o o . . o o o . . . . + . . o . + . + . o o o . . o o . . o o o o . + . + + +
. . . . . . . + . . . . . o . o o o o o o o o o . o . o o . . . . . + . . . . o . . . . . . . . . . o . o . . o . o . . + + + +
. o . o . o . . . . o o o . o o o o o o o o o . o o . o . . . . . . . . o . . o o . . . o o . . . . . . . . . o o . . + . . + +
. o o o o o o o o . . o . o o o . . . . o . o o o o o . . + . + . . o . . o . . o o o o . o . o . . + . . . o o . o . . . . . +
. o o . o o o . . o . o o o o o o . o . . o o o o o . . . . . . . o . . . . . . . . o o . o o . o . + . o o o . . . o . . . + +
o . o o o o o o . o o o . . o . o . o o o o o o o o . + + . . o o . . . . . . . + . o o o o . . . . + . . o . o . . . o . . . .
. . o o o o o o . o o . . . . . . . o . o o o o o o . . . . o . o o . . . . o . . . o o o o . . . . + . . . . . o . . o o . . o
+ . . . o o o o o . . o . . . o . o o o o . . . o o o . . o . . o o . + . o o . o o . . o o . . + . . . + . . . . . o o o o o o
. . . . . o . . o . . . . . . o o . o o o . . . . . o o o o o . . . . . . . . . . . . . . . . + . + . . + + + + . . o o o o . .
. . . o o o . . . o o o o o . o o . . o o . + . o o o o . . . . . o o . . + + . + . . o . o . . . . . + + . + + . . . o o o . o
+ + . . . o o . o o o o o o . o . . . . . . + . . . o . o . . o o . . . . . . . . + . . . . . . + . + . + + + + + . . . o o o .
+ + . + . . . o o o . . o . o . . . . + + + . . . . o o o . . . o o . . o . o o . . . . . . + . . . + . + + + . . . . . . . o .
+ + . . . o o o o . . o . o o o o . . + + . + + + . o o o o o . . o . . . . o o o . + + . . . . . . . . . . . . o o . . o o . .
+ . + . . o o o o . . . o o . . o . . . . . . . . . o o . o o . . . . . . o . o . . . + . . . . . . . o . . o . o . . o . . . .
+ + . . o o o o . . + . . . o . . o o . . . o . . . o o o o . . . . . . . o o . . . . . . + . o . o . . o o o . o o o . . . . o
. + . . . . o o o . . . . . o . o o . o . . . . . o . o . . . . o . . . . o . . . . . + + + . o o . o o o o . . o o o o o o . .
. . . o o o o o . . + . . . o . o . . . . . . o o o . o . o . . . . . . . o o . . . . . . . . . o . . . o . o . o o . o . o o .
. . o . o o o . o . . . o o o o o o . + . . o o o o o o . o o . . . . . . . . . . . . . . . . o . . . . . . . o . . o . . . . o
+ . o . . o . o o o . . o . o o o o . . . . . o o o o . . . . o o o o o . . . o . . . . o o . o . . . . . . . . . . o o . . . o
. . o . . o o . . . o o o o o o o . . . . . o . . . . . + . . o o . o . . + . . . . . . . o o . o o . . o . + + + . . o . . . o
o o o o . o o o o . o o . . o o o . o . + . o . . o . . + . . . o o o . + + . + . . o . . . . o . . . . o . . + . . . o o . o o
o o o . . . o o o o o . . . o o . o . . . . o . o o . + . + . . o o o . . . . + . . . . . . o . . . . o . . . + . . . o o o o o
. . . . o . o . . o . . + . . . o o o . . . . o o o . . . . . o o . o . o . . . . + . . o . . . . . o . . . . . . o . . o o o o
+ + . . . . . . o . o . . . o o . . o . . . . o . o o . o o . . . o o . o o . . . + . . . . . . . o . . . + . . . . o o o o o o
+ + . + . . . . . . o . + . . . . o . . . . o o . o . . o o o . o o o o o o o o . . . o . . . o o o o . . . . . o . . o o o o o
+ . . + . + . + . . . . + + . o . . . + . . . o . . . . . o . . o o o . o o o o o . . . . . o . o . . o o . . o o . . . o . o .
+ + + + + . . + . + . + . + . . . . . . . . o o . . . . o . . . . . . . . . o o o o o o o . . . . . o o o o o o . . . . . . o o .
+ . . . . + . + + . + . . . . . + . . o . . . o . . o . . . . + . . o . . . . o o o o o . . + . o . o o . . . . . . o . . . o o
. . o . . . . . . . . . o . . + . . . . o . . . . . . . . . . . . o . o o o o . o o o o o . . . . . o . . + . . . . . o o o o .
. o o o . o o o o o . . o o . . . . . o . o . . + . + . + . o o o o . . o o o o o o o o o o o o . o . . . . . . . . o . . . o o
. . . . . o o o o . . . o . o o . . . . . . . . + + + . . . . . o o o o o o o . o o o o o o o . o . o . . . . + . . . . . . o o
+ + . . . o . o . . o o . o . . . . o o o . . . . . . . + . + . . o o o . o . . . o o o o o o o . o . . o o . . . . . . . . o o
. + + . . . . . . . o o . . . o . o o o o . . . + . . . . . . . . . o o o o . . . o o o o o . . o o . . o o o o . + . . . . . .
+ + + + + + . . . . . . o . . . . . o . o o o . . . o . o . o . . . . o o o . + . . . o o o . . . o . . o o o . . + . + . + . +
+ + + + + . . . . + . . . o . + . . . o o o o o o . . o . . . o . + . . o . . + . . o . . . . . . . . o o . . o . . . . + . + + +
+ + + + + + + + + + . . o . . . . . . o o o o . o o . . . + . o . . . o o o . . + . . . . . . . . . o . . . . o . o . . + + + +
+ + + . . + + + + . . o . o . o o . o o . . . o . . . + . . . . o . o o o o o . . . . . o . o o o . o o . . . . . . . + + . . .
. . . + . + + + + . . . o . . o . . . o . + . . o . . . . + . . . . o . . o o o . o . . . . o o . . . o . . . + . . + + . . o .
+ . . . + + . + + + + . o o o o . + . . . . . . . o o o . . . . . o . . o o o . o . . . + . o o o . o o o o . . + + + + . o . .
+ + + . . + + + + + . . . . . o . + . + . . o . . . . o . o . . . . . . o o o . . . . . . . . o . o . o . . . . + . + + . . . .
+ + + + + . . + + + + . + + . o . + + + + . o . . + . o . . . . . + . . o . o . . o . . . . o o o o . o o . . + + . + + + + . .
```

(e) Difficulties Encountered and Conclusion

Dealing with deadlocks was a bit complicated since the “checkered” approach was chosen to split the processes. MPI does not come with spooling capabilities, meaning that the sender has to wait until the receiver receives, and the receiver has to wait until the sender sends. Another difficulty was debugging. After obtaining the correct result with the first input, the other inputs failed. And because of parallel programming, the debugging phase was tougher than usual. As the source of error might be anywhere and the processes should proceed at the same time.

In conclusion, parallel computing is very important nowadays as it lets the computer execute code more efficiently. This saves time, especially in complex programs. However, it is a complex task as it requires thorough and careful design decisions. Despite that, large and complex projects which need speed and accuracy greatly benefit from parallel programming.