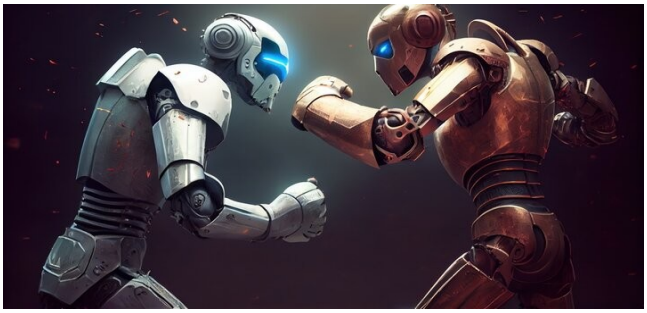
Date handed out: 29th of March, 2PM
Date submission due: 15th of April, 11 PM

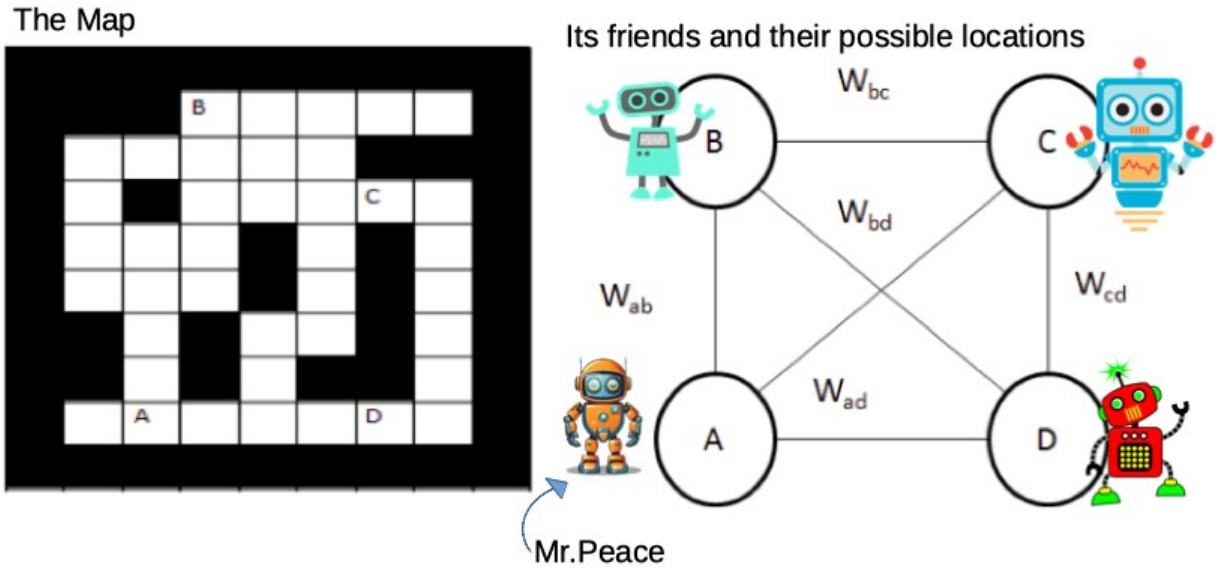
The overall goal of this assignment is to enable you practice how to use search algorithms we learnt in the class, all the pseudo-code will be provided to you and you must follow it for your implementation.

Important: All the material included in this assignment could be included in future quizzes, midterms and exams. I expect of both team members to understand each other's tasks and the total code despite the work load distribution.

Task 1: Long Lost Friends

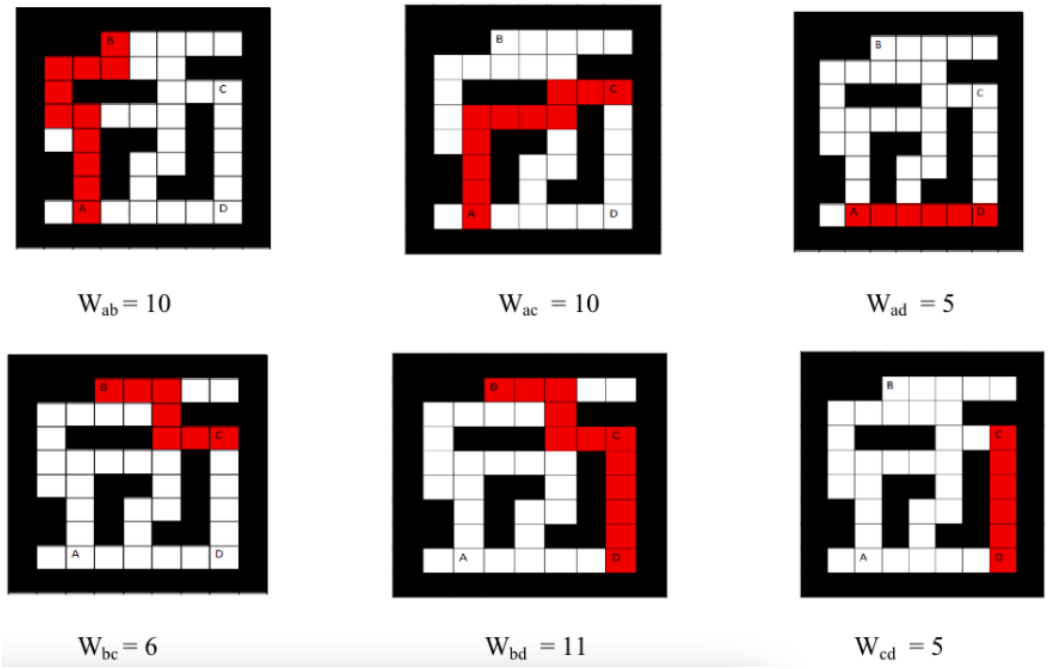


Context: It is 2124, and the robot war has started. You designed a friendly robot called Mr.Peace that decided to gather all its friends to fight the evil robots that want to control planet earth. Mr. Peace knows the locations of its friend and has a map to follow You must pose a search problem whose solution is an all-purpose sequence of actions such that, after executing those actions, the friends will be reunited.

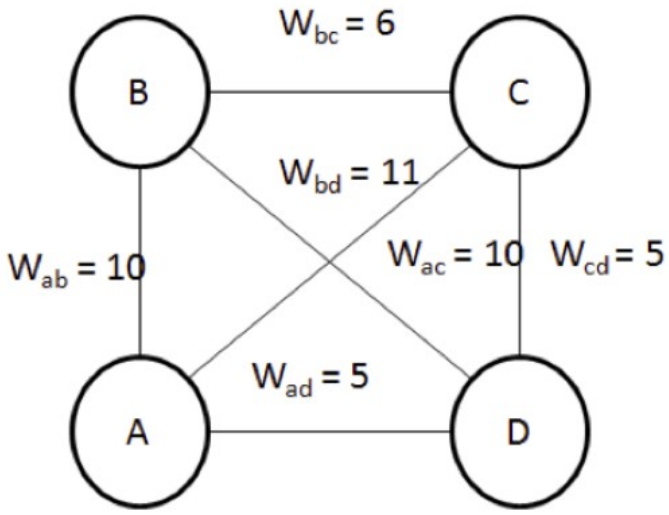


Problem 1:

Mr.peace has limited resource and thus, it will try to construct a graph space out of the map by identify the shortest path between each two friends, in other words, the shortest path between each two locations A → B (W_{ab}) and B → C (W_{bc}) and so on. Mr.Peace initial positions is A, and it can move only in four directions: Up, Right, Down, Left in the white boxes and not in the black ones with move cost equal to 1, the shortest paths are:



assuming that the graph is fully connected – meaning that ALL locations are reachable from all other locations. Give the original map, and the above illustration, the shortest path graph should be:



To construct this graph, it should first use A* with Manhattan distance between two points as a heuristic. A* pseudocode is:

```
Algorithm      A* search
Step 1. Let Q be a queue of partial paths (initially
root to root, length 0);
  Step 2. While Q is not empty or failure
      Step 2.1 if the first path P reaches the goal
      node then return success
      Step 2.2.remove path P from the queue;
      Step 2.3 extend P in all possible ways and add
      the new paths to the queue
      Step 2.4 sort the queue by the sum of two values:
      the real cost of P until now (g) and an estimate
      of the remaining distance (h);
      Step 2.5 prune the queue by leaving only the
      shortest path for each node reached so far;
Step 3. Return the shortest path (if success) or
failure.
```

Request for Problem 1:

1. Use a proper data structure to represent and construct the map given to you in form of a graph.
2. Your code should accept file\s, and from the file\s your program should be able to construct a graph of the map (or you can see it as a maze). Feel free to check the literature.
3. Each line contain three columns: node1, node2 and shortest distance. Each column is separated by a comma. Print the results in alphabetic order. Example:

a,b,10
a,c,10
a,d,5
b,c,6
b,d,11
c,d,5

Problem 2:

After determining the shortest path (now you have a space graph with minimal distance between locations), Mr. Peace needs to find a tour that allows it to visit all its friends exactly once and returning to the starting location (A) with minimal distance. To implement this task, you must use the graph and apply another search algorithm.

Request for Problem 2:

1) Apply Uniform Cost Search (UCF) and Depth First Search (DFS), pseudocode is available below for you as well.
2) The output should be a List the names of the nodes in the tour: Example (just for illusion): A – B – C - D - E - F - A

Algorithm Depth first search

```
Step 1. Form a queue Q and set it to the initial state
(for example, the Root).
Step 2. Until the Q is empty or the goal state is found
do:
            Step 2.1 Determine if the first element in
            the Q is the goal.
            Step 2.2 If it is not
                  Step 2.2.1 Remove the first element
                  in Q.
                  Step 2.2.2 Apply the rule to generate
                  new state(s) (successor states).
                  Step 2.2.3 If the new state is the
                  goal state quit and return this state
                  Step 2.2.4 Otherwise add the new
                  state to the beginning of the queue.
Step 3. If the goal is reached, success; else failure.
```

Algorithm Branch and bound (uniform cost) search

```
Return a solution or failure

Q is a priority queue sorted on the current cost from
the start to the goal
Step 1. Add the initial state (or root) to the queue.
Step 2. Until the goal is reached or the queue is empty
do
      Step 2.1 Remove the first path from the queue;
      Step 2.2. Create new paths by extending the first
      path to all the neighbors of the terminal node.
      Step 2.3. Remove all new paths with loops.
      Step 2.4. Add the remaining new paths, if any, to
      the queue.
      Step 2.5.   Sort the entire queue such as the
least-cost paths are in front.
End
```

Allowed programming languages to be used: Python, C other languages are not accepted.

Grading:

Grading	Points
Readme.txt (Your code should include a readme.txt that has a short description of your program and	10 (you cannot get this if you do not submit a code)

explains how to compile and run your code, please include your name, surname and id at the top)	
Problem 1, Data structure and reading the map from a txt file + representation of the map in the file (your code should not be restricted to the given map. It must be able to read any given map if we followed your representation)	10
Problem 1, A* and the construction of the graph (your program should clearly document how you solved the shortest path problem with the given heuristic)	20
Problem 2: UCS	15
Problem 2: DFS	20
Your output clearly shows the shortest paths for graph and the tour	10
Readable, clear code (with clear and detailed comments)	15