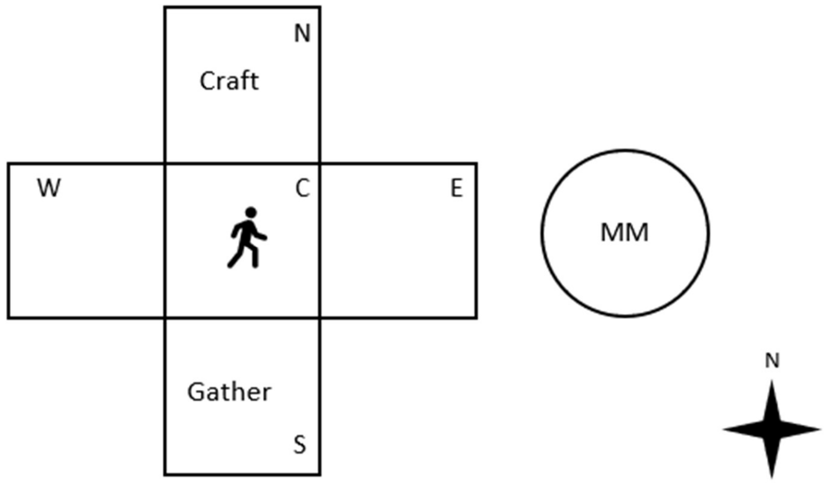
PART 2 : Value Iteration Algorithm

TASK 1:

Indiana Jones (IJ) is on an adventure to stop the **Mighty Monster (MM)** before it is too late. He has traveled for days and now he has found himself face to face with MM. He is in a room shaped like a plus **(+)** with only **5 positions** for him to stand. The squares are **Center (C), North (N), South (S), East (E), and West (W)**. MM is on the **East** side of the room. Indiana has infinite health (obviously), however, MM has only **max health of 100**. The episode will end once MM has **0 health** and Indiana will receive a **reward of 50.**

Indiana has a bow and a blade to attack the MM. Indiana at any moment can only carry **3 arrows (at max)**. Indiana can at max carry only **2 materials**.



Indiana's actions are dependent on his position in the room. At each timestep, he can take only **1 action**.

Center Square:
IJ can move: **{up (move to North Square), down, left, right, stay}** and action will be successful only **85%** of the time; the other times, he will move to the **East** Square. IJ can **shoot**, which will cause him to **lose 1 arrow.** Because this is a dark room, Indiana's accuracy is worse than before. The arrow will hit MM with a probability of **0.5** and miss otherwise. If it hits MM, it will deal **25 damage**. IJ can also **hit with his blade**, which will make contact with MM with a probability of **0.1** and miss MM with a probability of **0.9** If it hits MM, it will deal **50 damage**.

North Square:
IJ can move: **{down (move to Center Square), stay}** and the action will be successful only **85%** of the time; the other times, he will teleport to the **East** Square. IJ can **craft arrows** here if he has **1 Material**. Upon crafting, IJ will lose his Material and gain **1 arrow** with a probability of **0.5**, **2 arrows** with a probability of **0.35**, and **3 arrows** with a probability of **0.15**.

South Square:
IJ can move: **{up (move to Center Square), stay}** and the action will be successful only **85%** of the time; the other times, he will teleport to the **East** Square. IJ can **gather material**. He will **gather successfully** with a probability of **0.75** and fail with a **probability of 0.25**. Upon gathering **successfully**, IJ will gain **1 Material**, if he does not have material, else there will be no change.

East Square:
IJ can move: **{left (move to the Center Square), stay}**, and the action will be successful **100%** of the time. Indiana can **shoot arrows** here. The arrow will hit MM with a probability of **0.9** and miss MM with a probability of **0.1**. The arrow will deal **25 damage** to MM. Indiana can also **hit** with his blade. He will make contact with MM with a probability of **0.2** and miss with a probability of **0.8**. The blade will deal **50 damage** to MM.

West Square:
IJ can move: **{right (move to the Center Square), stay}**, and the action will be successful **100%** of the time. Indiana can **shoot** arrows here. The arrow will hit MM with a probability of **0.25** and miss MM with a probability of **0.75**. Upon hitting MM, the arrow will deal **25 damage** to MM.

At intervals, MM can also attack IJ. MM will indicate that he plans to attack soon by entering **Ready State**. MM enters the **Ready State** from **Dormant State** with a probability of **0.2** at each step or remains in the **Dormant State** with a probability of **0.8**. When MM is in the **Ready State**, he may **attack** in the timestep with a probability of **0.5** and enter the **Dormant State** or not attack with a probability of **0.5** and remain in the **Ready State**. When MM attacks he will only affect Indiana if IJ is either on the **East Square or Center Square** at that time step. In this case, IJ will **drop all his arrows** (if he has any), and MM will **regain 25 health**. If Indiana had planned to take any action at this step, it would be **unsuccessful**. Indiana will also get a **negative reward** of **-40.** *The probabilities for the transitions will scale appropriately with this new condition added.*

Since Indiana needs to stop MM asap, there is a penalty for each timestep. The penalty is given by:

```
Step Cost = -10/Y
```

```
arr = [1/2, 1, 2 ]
Y = arr[X mod 3]
X: team number
```

Parameters:

- `Gamma = (Discount Factor) 0.999`
- `Delta = (Convergence or Bellman error)` 10^{-3}

Output the trace of the state, the chosen action at that state and the value of state for all iterations from 0 till convergence. (output format on next page)

In the report:

- **Interpret and comment on the results in the trace file. Analyze the choice of actions for the states, rate of convergence, and any other meaningful pattern.**
- **Assume the start states as (given below) and simulate the game to find the set of actions till the end state (choose actions based on the policy). Print the order of the actions, the state transitions and comment on the results. No specific submission format.**
    1. **(W, 0, 0, D, 100)**
    2. **(C, 2, 0, R, 100)**

TASK 2:

Case 1:
- Indiana now on the LEFT action at East Square will go to the West Square.

Case 2:
- The step cost of the STAY action is now zero.

Case 3:
- Change the value of gamma to 0.25

Output the trace for each of the cases and comment on the results in the report. Justify the change in policy for each case. Note that for each case, every other parameter except the ones mentioned remain unchanged from task 1.

Submission format of trace files:

```
For each iteration:
  print "iteration=<iteration_number>"
  For each state:
    print "(<pos>,<mat>,<arrow>,<state>,<health>):<action>=[<state_value>]"
```

The values of some of the variables are restricted as follows:

```
● <pos> : {W, N, E, S, C}         // position of IJ
● <mat> : {0, 1, 2}               // material with IJ
● <arrow> : {0, 1, 2, 3}          // number of arrows
● <state> : {D, R}                // ready and dormant state of MM
● <health> : {0,25,50,75,100}     // MM's health
● <action> : {UP, LEFT, DOWN, RIGHT, STAY, SHOOT, HIT, CRAFT, GATHER,
  NONE}
```

Notes:

- `NONE` action is for absorbing states with 0 state value
- State values to be printed rounded to 3 decimal place
- Iterations to be printed from 0 till convergence
- For each iteration, ensure to list each state (order of states is flexible)
- Language allowed: `Python3`, Library allowed: `Numpy`

Example trace file:

```
iteration=0
(W,0,0,D,0):RIGHT=[12.231]
(W,0,0,D,25):RIGHT=[6.000]
.
.
.
(C,2,3,R,100):SHOOT=[1.232]
iteration=1
(W,0,0,D,0):STAY=[8.999]
.
.
.
.(so on until the end of iterations)
```

Marking Scheme:

- This part of the assignment is for 50 marks.
    - Task 1 : 23
    - Task 2: 27 (9 for each case)

PART 3 : Linear Programming (LP)

You are going to be solving the same problem in part 2 task 1, but here the problem will be formulated as a LP and you will be making use of the `cvxpy` library to solve it. It is a good idea to get familiar with this library first. A sample will soon be posted on moodle using this library to solve a LP.

There are, however, a few changes from the previous part:

- When MM's health reaches 0, IJ quest finished but gets **NO reward**. Otherwise use the same penalty (as your part 2) for other actions

Use the cxvpy library to solve and determine the policy for the LP.

In the report:

- **Explain procedure of making A matrix**
- **Explain procedure of finding the policy and analyze the results.**
- **Can there be multiple policies? Why? What changes can you make in your code to generate another policy? (Do not paste code snippets, explain the changes in terms of how they will affect the A matrix, R vector, alpha vector etc.)**

Output Format:

You are required to output 6 things. You are required to put all of them in a dictionary with keys as specified:

Description	**Data Type**	**Dict Key**
The A matrix	List of lists	`a`
The R array	List of floats	`r`
The alpha array	List of floats	`alpha`
The optimised x array	List of floats	`x`
The derived deterministic policy	List of lists; Each sublist is of the form [state, action] similar to format of previous part	`policy`
The finalized objective value	float	`objective`

Note:

- The A matrix should consist of **only the valid actions for each state**. Eg, for states in which IJ has 0 arrows, "SHOOT" is an invalid action and therefore should not appear in the A matrix, nor in any of the alpha, R or X vectors.
- The number of rows (lists) in A matrix must be equal to the **number of states**. The dimensions of other values should follow logically from this.
- **Ignore self-looping transitions of a valid action** to avoid unbounded LP. For example, suppose IJ had 0 materials and performs the action `gather`, then there is a probability of 0.25 for the state to not change, and therefore you wouldn't add this probability to the A matrix. In this case, you would only consider the 0.75 probability of reaching the state with one extra material.
- Language allowed: `Python3`, Library allowed: `Numpy, cvxpy`
- The calculated values should be accurate to 3 decimal places.
- You need to json dump (using `json.dump` in python) the dictionary into the `part_3_output.json` file. Please check that you are able to read back the exact same dictionary using `json.load`, before submitting.

Sample Dictionary:

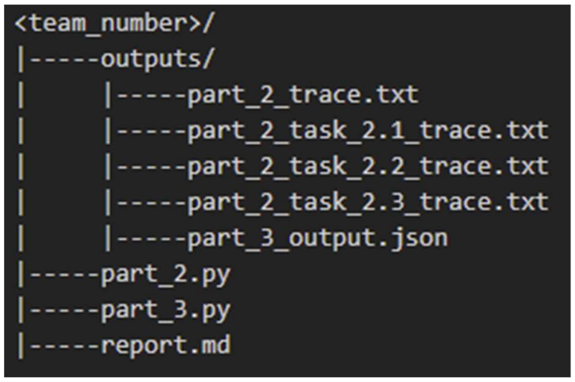
```
{
     "a": [[1.0, 0.0, 0.0, 0.0],
           [0.0, 1.0, 0.0, 0.0],
           [0.0, 0.0, 1.0, 0.0]],
     "r": [1.0, 1.0, 1.0, 1.0],
     "alpha": [0, 0, 0, 1],
     "x": [1.724, 1.784, 0.645, 1.854],
     "policy": [[(W,0,0,D,0), "RIGHT"],
                [(W,0,0,D,25), "RIGHT"],
               ...
                [(C,2,3,R,75), "HIT"],
                [(C,2,3,R,100), "SHOOT"]],
     "objective": -11.54321
}
```

Marking Scheme:

- This part of the assignment is for 30 marks.
  - Code : 12
  - Automated evaluation output: 12
  - Report : 6

Submission Details

- Part 1 needs to be submitted handwritten. Separate submission portal will be available to submit the scanned pdf.
- For part 2 and 3 submit a single zip file named **<team_number>.zip** with the following directory structure when unzipped:



```
<team_number>/
|-----outputs/
|      |-----part_2_trace.txt
|      |-----part_2_task_2.1_trace.txt
|      |-----part_2_task_2.2_trace.txt
|      |-----part_2_task_2.3_trace.txt
|      |-----part_3_output.json
|-----part_2.py
|-----part_3.py
|-----report.md
```

- Executing the `part_2.py` should create the `part_2_trace.txt` in the output folder. Similarly, the `part_3.py` should create the `part_3_output.json`
- Part of the evaluation will be automated, so kindly stick to the output and submission format. Failure to stick to the format will result in a direct 0 for the respective parts.
- Plagiarism will be penalized.
- **Deadline for the assignment is 5th April 23:59**