first name **last name** **id#**

1. [**5 marks**] For this sliding tile position, give A) the number of inversions and B) the Manhattan (taxicab) heuristic score. Also, from this position (the root), draw the next two levels of the search space graph. Finally, give C) the number of positions reachable from this position, i.e. the number of nodes in this component of the search space graph.

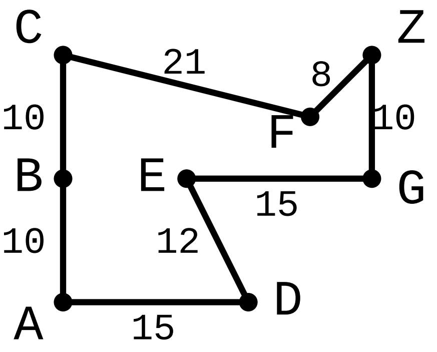
```
1 3 5 7         A ____        B ____        C ____
2 4 6 -
```

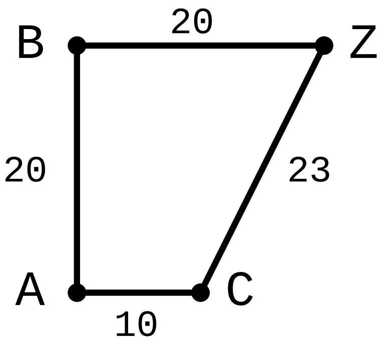
2. [**3 marks**] `15puzzle.py` allows the user to give intermediate destinations for the sliding tile breadth-first-search. E.g. with schedule A) below, the 1st destination has tiles 1,2,3,4 in their correct final positions; with schedule B) the 1st destination has tile 1 in its correct final position. With input (15 14 13 12 10 9 8 11 7 6 4 2 5 1 3), I ran the program three times: once with each schedule. Below, for each run, give the number of moves in the solution found, and the total numbers of nodes searched.

   Hint: each answer is in {82, 90, 120, 6865, 145722, 1765263 }.

```
                                                                 moves   nodes searched
A) [[1, 2, 3, 4], [5, 9, 13], [6, 7, 8, 10, 11, 12, 14, 15]]     _____       _____
B) [[1], [2], [3, 4], [5], [6], [7, 8], [9, 13], [10, 14], [11, 12, 15]   _____       _____
C) [[1, 2], [3, 4], [5, 6, 7, 8], [9, 10, 11, 12, 13, 14, 15]]   _____       _____
```

3. [**8 marks**] Here are two roadmaps. Each edge label is a road distance. Below are heuristic estimates of distance remaining to Z, and A* pseudocode from class. Trace the pseudocode on the graph at left, with start A and finish Z. Each time `current` is assigned a node, give the node and its priority. For the graph at right, this is the answer: A 0, C 32, Z 33





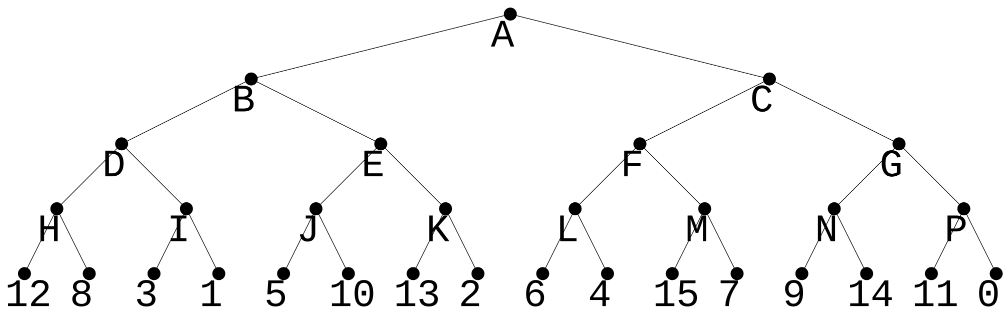
heuristic						
BZ	CZ	DZ	EZ	FZ	GZ	ZZ
26	24	22	18	7	10	0

example graph heuristic		
BZ	CZ	ZZ
20	22	0

`answer: __________________________________`

`example graph answer: A 0, C 32, Z 33`

```
fringe = PQ()
fringe.add(start, 0)
parent, cost, done = {}, {}, []
parent[start], cost[start] = None, 0
while not fringe.empty():
  current = fringe.remove() # min priority
  done.add(current)
  if current == target: break
  for next in nbrs(current):
    if next not in done:
      new_cost = cost[current] + wt(current, next)
      if next not in cost or new_cost < cost[next]:
        cost[next] = new_cost
        priority = new_cost + heuristic(next, target)
        fringe.add(next, priority)
        parent[next] = current
```



4. [**2 marks**] For each node in the tree above, give the node's minimax value: assume that the root is a min node.

```
A ___ B ___ C ___ D ___ E ___ F ___ G ___ H ___ I ___ J ___ K ___ L ___ M ___ N ___ P ___
```

5. [**3 marks**] Now assume that the root is a max node. Rearrange (permute) the leaf nodes so that the root minimax value is as small as possible. Show the rearranged leaf values (from left to right) and each node's minimax value.

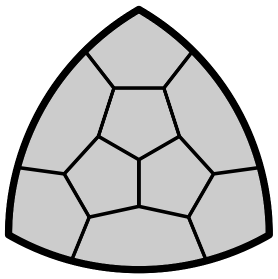
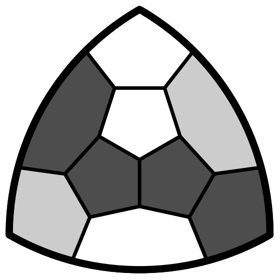
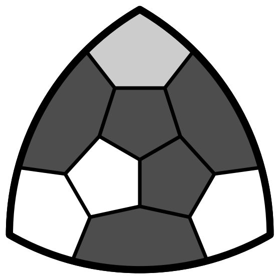
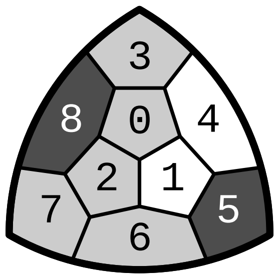
```
rearranged leaf values ___ ___ ___ ___ ___ ___ ___ ___ ___ ___ ___ ___ ___ ___ ___ ___

A ___ B ___ C ___ D ___ E ___ F ___ G ___ H ___ I ___ J ___ K ___ L ___ M ___ N ___ P ___
```

6. [**3 marks**] This python function computes minimax values of complete binary tree in bottom-up order. It takes as input the list L of leaf nodes and whether the root is a max-node or a min-node and returns the root minimax value. (As in the previous question, a complete binary tree has exactly 2^t nodes at each level t, starting with 1 node at level 0.) Give the missing function code fragments a),b),c),d).

```
def bottom_up_mmx(L, is_max):
  if len(L) == 1:
    return(L[0])
  M = []
  for j in range(len(L)//2):
    x, y =  2*j ,  ___(a)___
    M.append(max(L[x],L[y]) if is_max else  ___(b)___
  return bottom_up_mmx( ___(c)___, ___(d)___  )
```

```
(a) _______________    (b) _______________    (c) _______________    (d) _______________
```

7. [**4 marks**] Similar to Hex, Trex is a two-player (Black, White) alternate-turn game played on the 9-cell board above left. On a turn, a player colors an empty cell. The winner is whoever forms a connected group joining all three sides. E.g. Black has won the two finished games above. Notice that each corner cell touches two sides.

   Consider the position above right. Labels are cell numbers. Black plays next. White has a win-threat: play at 7 and join the group at {1,4} with a virtual connection. In sorted order, give the cells in the Black mustplay region formed by this White win-threat. Answer like this: 0 4 6 8.

   `answer: ___________________________    justification:`

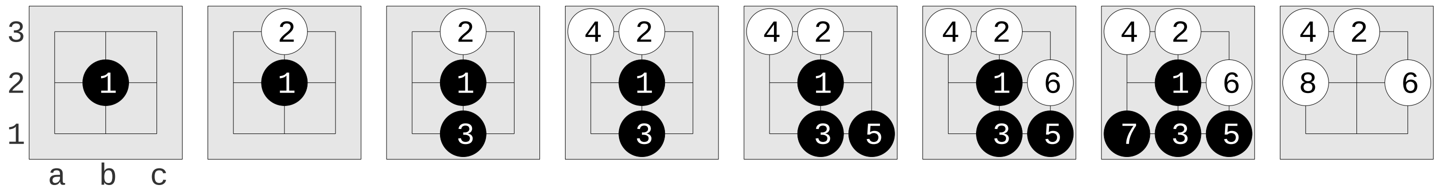
   In sorted order, give all winnning Black moves. Answer like this: 1 3 5 6 8 (or "none").

   `answer: ___________________________    justification:`

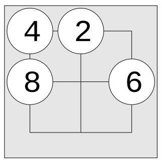
8. [**4 marks**] Vim (like Nim) is a 2-player alternate-turn game. There are **3** piles of stones. On a turn, if some pile has at least one stone, then player-to-move must take **1 or 2 stones** from one pile; if player-to-move has no legal moves, then they **win**. We describe Vim position by giving each pile size, in sorted order. E.g. X and Y start from position (1,2,7). X moves first and has these move options: take 1 stone from the 1-pile, leaving (0,2,7); take 1 or 2 stones from the 2-pile, leaving (1,1,7) or (1,0,7)=(0,1,7); take 1 or 2 stones from the 7-pile, leaving (1,2,6) or (1,2,5). Here is their game. (1,2,7); X to (1,2,5); Y to (1,0,5)=(0,1,5); X to (0,1,3); Y to (0,1,2); X to (0,1,0)=(0,0,1); Y to (0,0,0): X wins, Y loses.

   A, B, C, D, E, F are positions (0,0,0) (0,0,1) (0,1,3) (1,1,2) (1,2,4) (1,2,6) respectively. For Vim, for each of these positions, give the player-to-move minimax value: **win** or **loss**. The first answer is given for you.

   `A: win   B: _____    C: _____    D: _____    E: _____    F: _____`



9. **[4 marks]** After these moves in a go game (Tromp-Taylor, no-suicide, positional superko), each player follows a minimax strategy. Give the final score: `(B stones+territory ____ W stones+territory ____)`.

   From here, draw the next two levels of a proof tree that confirms your answer. Below each bottom node in your diagram, give the final minimax score from there. 

10. **[4 marks]** Modify rock-paper-scissors so that Rose's payoff matrix is shown below. The game is 0-sum: Colin's payoff is the negative of Rose's. E.g. if Rose plays rock and Colin plays paper then Rose loses 2 points and Colin gains 2 points. (A) What is Rose's expected payoff if she plays the mixed strategy (r 1/3, p 1/2, s 1/6) and Colin plays the pure strategy rock? (B) Repeat (A) when Colin plays paper. (C) Repeat (A) when Colin plays scissors. (D) The minimum of your answers to (A),(B),(C) is Rose's guaranteed expected payoff: she will alway win at least this amount. Give a mixed strategy for Rose that maximizes her guaranteed expected payoff.

```
   rock paper scissors     (A) _____
r    0   -2    1           (B) _____
p    2    0   -1           (C) _____
s   -1    1    0           (D)      (r ____, p ____, s ____)
```

1. [**5 marks**] For this sliding tile position, give A) the number of inversions and B) the Manhattan (taxicab) heuristic score. Also, from this position (the root), draw the next two levels of the search space graph. Finally, give C) the number of positions reachable from this position, i.e. the number of nodes in this component of the search space graph.

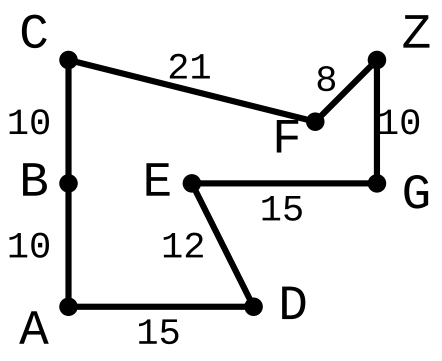
```
1 3 5 7        A ____        B ____        C ____
2 - 4 6
```

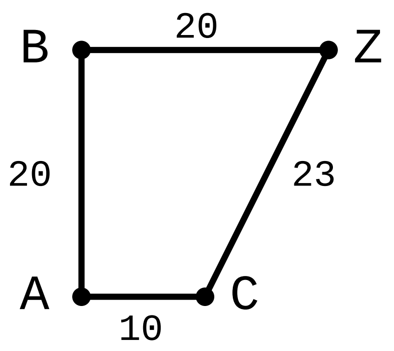
2. [**3 marks**] `15puzzle.py` allows the user to give intermediate destinations for the sliding tile breadth-first-search. E.g. with schedule B) below, the 1st destination has tiles 1,2,3,4 in their correct final positions; with schedule C) the 1st destination has tile 1 in its correct final position. With input (15 14 13 12 10 9 8 11 7 6 4 2 5 1 3), I ran the program three times: once with each schedule. Below, for each run, give the number of moves in the solution found, and the total numbers of nodes searched.

   Hint: each answer is in {82, 90, 120, 6865, 145722, 1765263 }.

```
                                                                    moves   nodes searched
A) [[1, 2], [3, 4], [5, 6, 7, 8], [9, 10, 11, 12, 13, 14, 15]]      _____       _____
B) [[1, 2, 3, 4], [5, 9, 13], [6, 7, 8, 10, 11, 12, 14, 15]]        _____       _____
C) [[1], [2], [3, 4], [5], [6], [7, 8], [9, 13], [10, 14], [11, 12, 15]] _____  _____
```

3. [**8 marks**] Here are two roadmaps. Each edge label is a road distance. Below are heuristic estimates of distance remaining to Z, and A* pseudocode from class. Trace the pseudocode on the graph at left, with start A and finish Z. Each time `current` is assigned a node, give the node and its priority. For the graph at right, this is the answer: A 0, C 32, Z 33





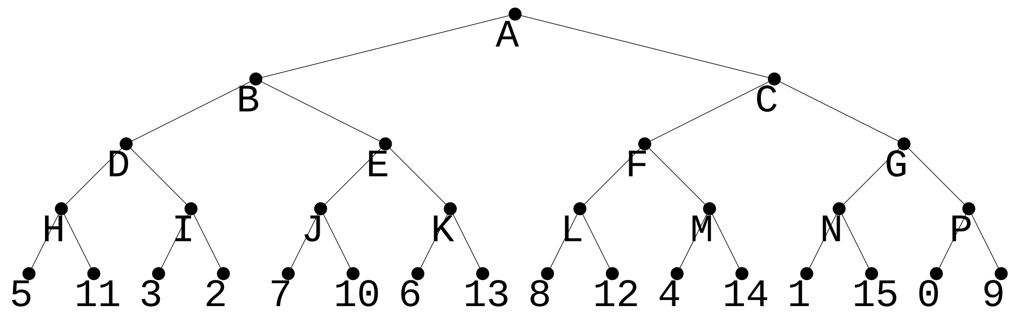
```
heuristic
BZ  CZ  DZ  EZ  FZ  GZ  ZZ
26  25  20  17   7  10   0

answer: __________________________________
```

```
example graph heuristic
BZ  CZ  ZZ
20  22   0

example graph answer: A 0, C 32, Z 33
```

```
fringe = PQ()
fringe.add(start, 0)
parent, cost, done = {}, {}, []
parent[start], cost[start] = None, 0
while not fringe.empty():
  current = fringe.remove() # min priority
  done.add(current)
  if current == target: break
  for next in nbrs(current):
    if next not in done:
      new_cost = cost[current] + wt(current, next)
      if next not in cost or new_cost < cost[next]:
        cost[next] = new_cost
        priority = new_cost + heuristic(next, target)
        fringe.add(next, priority)
        parent[next] = current
```



4. **[2 marks]** For each node in the tree above, give the node's minimax value: assume that the root is a min node.

```
A ___ B ___ C ___ D ___ E ___ F ___ G ___ H ___ I ___ J ___ K ___ L ___ M ___ N ___ P ___
```

5. **[3 marks]** Now assume that the root is a max node. Rearrange (permute) the leaf nodes so that the root minimax value is as small as possible. Show the rearranged leaf values (from left to right) and each node's minimax value.

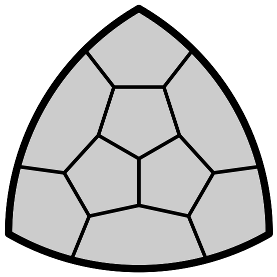
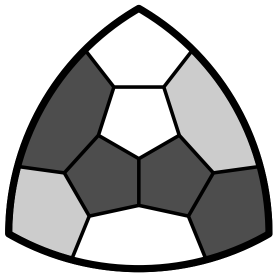
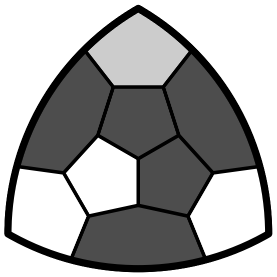
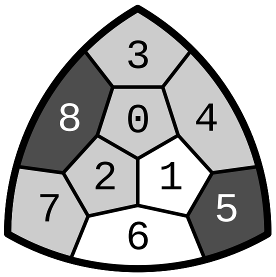
```
rearranged leaf values ___ ___ ___ ___ ___ ___ ___ ___ ___ ___ ___ ___ ___ ___ ___ ___

A ___ B ___ C ___ D ___ E ___ F ___ G ___ H ___ I ___ J ___ K ___ L ___ M ___ N ___ P ___
```

6. **[3 marks]** This python function computes minimax values of complete binary tree in bottom-up order. It takes as input the list L of leaf nodes and whether the root is a max-node or a min-node and returns the root minimax value. (As in the previous question, a complete binary tree has exactly 2^t nodes at each level t, starting with 1 node at level 0.) Give the missing function code fragments a),b),c),d).

```
def bottom_up_mmx(L, is_max):
  if len(L) == 1:
    return(L[0])
  M = []
  for j in range(len(L)//2):
    x, y =  2*j ,  ___(a)___
    M.append(max(L[x],L[y]) if is_max else  ___(b)___
  return bottom_up_mmx( ___(c)___, ___(d)___  )

(a) _______________    (b) _______________    (c) _______________    (d) _______________
```

7. **[4 marks]** Similar to Hex, Trex is a two-player (Black, White) alternate-turn game played on the 9-cell board above left. On a turn, a player colors an empty cell. The winner is whoever forms a connected group joining all three sides. E.g. Black has won the two finished games above. Notice that each corner cell touches two sides.

   Consider the position above right. Labels are cell numbers. Black plays next. White has a win-threat: play at 3 and join the group at {1,6} with a virtual connection. In sorted order, give the cells in the Black mustplay region formed by this White win-threat. Answer like this: 0 4 6 8.

   `answer: ____________________________     justification:`

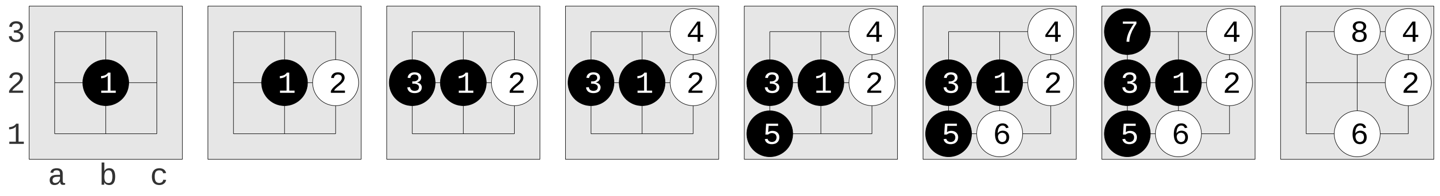
   In sorted order, give all winnning Black moves. Answer like this: 1 3 5 6 8 (or "none").

   `answer: ____________________________     justification:`

8. **[4 marks]** Qim (like Nim) is a 2-player alternate-turn game. There are **3** piles of stones. On a turn, if some pile has at least one stone, then player-to-move must take **1 or 2 stones** from one pile; if player-to-move has no legal moves, then they **win**. We describe Qim position by giving each pile size, in sorted order. E.g. X and Y start from position (1,2,7). X moves first and has these move options: take 1 stone from the 1-pile, leaving (0,2,7); take 1 or 2 stones from the 2-pile, leaving (1,1,7) or (1,0,7)=(0,1,7); take 1 or 2 stones from the 7-pile, leaving (1,2,6) or (1,2,5). Here is their game. (1,2,7); X to (1,2,5); Y to (1,0,5)=(0,1,5); X to (0,1,3); Y to (0,1,2); X to (0,1,0)=(0,0,1); Y to (0,0,0): X wins, Y loses.

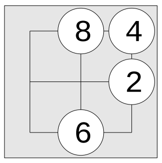
   A, B, C, D, E, F are positions (0,0,0) (0,0,2) (0,1,1) (1,1,3) (1,2,2) (1,2,7) respectively. For Qim, for each of these positions, give the player-to-move minimax value: **win** or **loss**. The first answer is given for you.

   `A: win   B: _____    C: _____    D: _____    E: _____     F: _____`



9. **[4 marks]** After these moves in a go game (Tromp-Taylor, no-suicide, positional superko), each player follows a minimax strategy. Give the final score: `(B stones+territory ____ W stones+territory ____)`.

   From here, draw the next two levels of a proof tree that confirms your answer. Below each bottom node in your diagram, give the final minimax score from there.



10. **[4 marks]** Modify rock-paper-scissors so that Rose's payoff matrix is shown below. The game is 0-sum: Colin's payoff is the negative of Rose's. E.g. if Rose plays scissor and Colin plays rock then Rose loses 2 points and Colin gains 2 points. (A) What is Rose's expected payoff if she plays the mixed strategy (r 1/3, p 1/2, s 1/6) and Colin plays the pure strategy rock? (B) Repeat (A) when Colin plays paper. (C) Repeat (A) when Colin plays scissors. (D) The minimum of your answers to (A),(B),(C) is Rose's guaranteed expected payoff: she will alway win at least this amount. Give a mixed strategy for Rose that maximizes her guaranteed expected payoff.

```
   rock paper scissors      (A) _____
r    0   -1     2           (B) _____
p    1    0    -1           (C) _____
s   -2    1     0           (D)      (r ____, p ____, s ____)
```

first name **last name** **id#**

1. [**5 marks**] For this sliding tile position, give A) the number of inversions and B) the Manhattan (taxicab) heuristic score. Also, from this position (the root), draw the next two levels of the search space graph. Finally, give C) the number of positions reachable from this position, i.e. the number of nodes in this component of the search space graph.

```
1 3 5        A ____        B ____        C ____
2 4 6
7 - 8
```

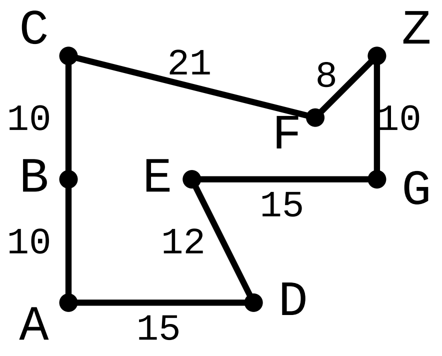
2. [**3 marks**] `15puzzle.py` allows the user to give intermediate destinations for the sliding tile breadth-first-search. E.g. with schedule C) below, the 1st destination has tiles 1,2,3,4 in their correct final positions; with schedule A) the 1st destination has tile 1 in its correct final position. With input (15 14 13 12 10 9 8 11 7 6 4 2 5 1 3), I ran the program three times: once with each schedule. Below, for each run, give the number of moves in the solution found, and the total numbers of nodes searched.

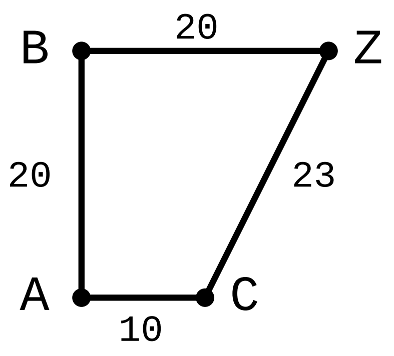
   Hint: each answer is in {82, 90, 120, 6865, 145722, 1765263 }.

```
                                                                 moves   nodes searched
A) [[1], [2], [3, 4], [5], [6], [7, 8], [9, 13], [10, 14], [11, 12, 15]] _____       _____
B) [[1, 2], [3, 4], [5, 6, 7, 8], [9, 10, 11, 12, 13, 14, 15]]           _____       _____
C) [[1, 2, 3, 4], [5, 9, 13], [6, 7, 8, 10, 11, 12, 14, 15]]             _____       _____
```

3. **[8 marks]**

Here are two roadmaps. Each edge label is a road distance. Below are heuristic estimates of distance remaining to Z, and A* pseudocode from class. Trace the pseudocode on the graph at left, with start A and finish Z. Each time `current` is assigned a node, give the node and its priority. For the graph at right, this is the answer: A 0, C 32, Z 33





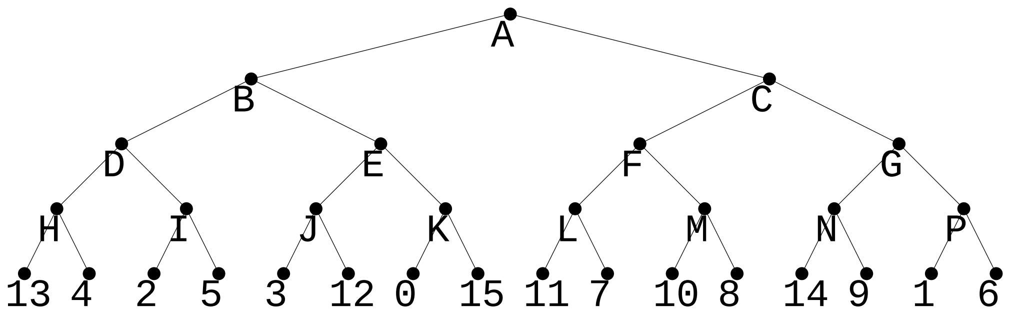
```
heuristic
BZ  CZ  DZ  EZ  FZ  GZ  ZZ
26  24  22  18   7   2   0

answer: __________________________________
```

```
example graph heuristic
BZ  CZ  ZZ
20  22   0

example graph answer: A 0, C 32, Z 33
```

```
fringe = PQ()
fringe.add(start, 0)
parent, cost, done = {}, {}, []
parent[start], cost[start] = None, 0
while not fringe.empty():
  current = fringe.remove() # min priority
  done.add(current)
  if current == target: break
  for next in nbrs(current):
    if next not in done:
      new_cost = cost[current] + wt(current, next)
      if next not in cost or new_cost < cost[next]:
        cost[next] = new_cost
        priority = new_cost + heuristic(next, target)
        fringe.add(next, priority)
        parent[next] = current
```



4. [**2 marks**] For each node in the tree above, give the node's minimax value: assume that the root is a min node.

```
A ___ B ___ C ___ D ___ E ___ F ___ G ___ H ___ I ___ J ___ K ___ L ___ M ___ N ___ P ___
```

5. [**3 marks**] Now assume that the root is a max node. Rearrange (permute) the leaf nodes so that the root minimax value is as small as possible. Show the rearranged leaf values (from left to right) and each node's minimax value.

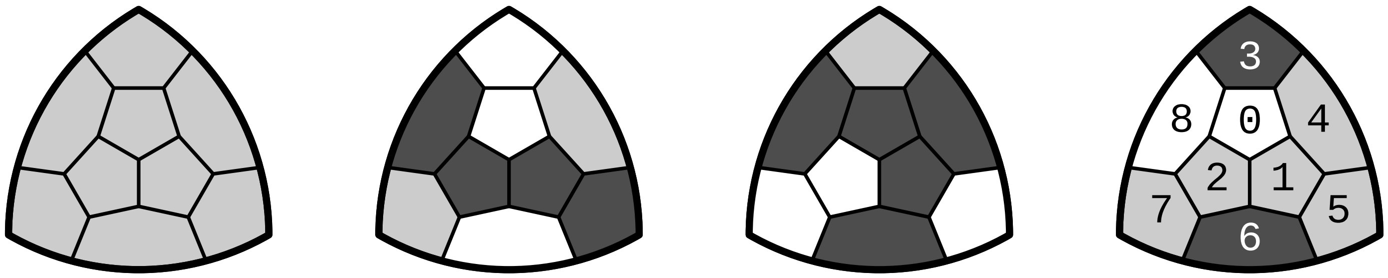
```
rearranged leaf values ___ ___ ___ ___ ___ ___ ___ ___ ___ ___ ___ ___ ___ ___ ___ ___

A ___ B ___ C ___ D ___ E ___ F ___ G ___ H ___ I ___ J ___ K ___ L ___ M ___ N ___ P ___
```

6. [**3 marks**] This python function computes minimax values of complete binary tree in bottom-up order. It takes as input the list L of leaf nodes and whether the root is a max-node or a min-node and returns the root minimax value. (As in the previous question, a complete binary tree has exactly 2^t nodes at each level t, starting with 1 node at level 0.) Give the missing function code fragments a),b),c),d).

```
def bottom_up_mmx(L, is_max):
  if len(L) == 1:
    return(L[0])
  M = []
  for j in range(len(L)//2):
    x, y =  2*j ,  ___(a)___
    M.append(max(L[x],L[y]) if is_max else  ___(b)___
  return bottom_up_mmx( ___(c)___, ___(d)___  )
```

```
(a) _______________    (b) _______________    (c) _______________    (d) _______________
```



7. [**4 marks**] Similar to Hex, Trex is a two-player (Black, White) alternate-turn game played on the 9-cell board above left. On a turn, a player colors an empty cell. The winner is whoever forms a connected group joining all three sides. E.g. Black has won the two finished games above. Notice that each corner cell touches two sides.

   Consider the position above right. Labels are cell numbers. Black plays next. White has a win-threat: play at 5 and join the group at {0,8} with a virtual connection. In sorted order, give the cells in the Black mustplay region formed by this White win-threat. Answer like this: 0 4 6 8.

   `answer: ____________________________    justification:`

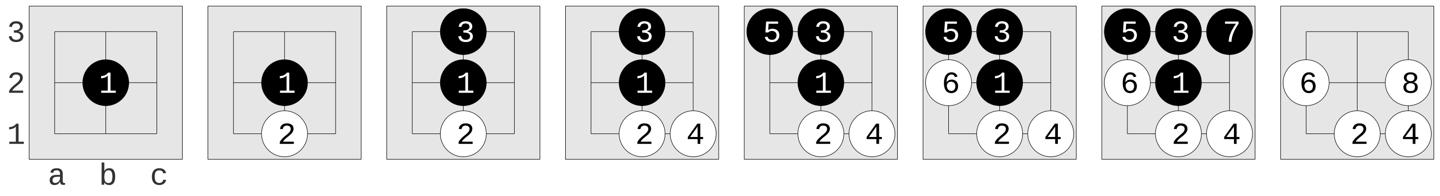
   In sorted order, give all winnning Black moves. Answer like this: 1 3 5 6 8 (or "none").

   `answer: ____________________________    justification:`

8. [**4 marks**] Zim (like Nim) is a 2-player alternate-turn game. There are **3** piles of stones. On a turn, if some pile has at least one stone, then player-to-move must take **1 or 2 stones** from one pile; if player-to-move has no legal moves, then they **win**. We describe Zim position by giving each pile size, in sorted order. E.g. X and Y start from position (1,2,7). X moves first and has these move options: take 1 stone from the 1-pile, leaving (0,2,7); take 1 or 2 stones from the 2-pile, leaving (1,1,7) or (1,0,7)=(0,1,7); take 1 or 2 stones from the 7-pile, leaving (1,2,6) or (1,2,5). Here is their game. (1,2,7); X to (1,2,5); Y to (1,0,5)=(0,1,5); X to (0,1,3); Y to (0,1,2); X to (0,1,0)=(0,0,1); Y to (0,0,0): X wins, Y loses.

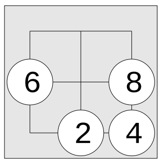
   A, B, C, D, E, F are positions (0,0,0) (0,0,3) (0,1,2) (1,1,1) (1,2,3) (1,2,5) respectively. For Zim, for each of these positions, give the player-to-move minimax value: **win** or **loss**. The first answer is given for you.

   `A: win   B: _____    C: _____    D: _____    E: _____     F: _____`



9. **[4 marks]** After these moves in a go game (Tromp-Taylor, no-suicide, positional superko), each player follows a minimax strategy. Give the final score: `(B stones+territory ____ W stones+territory ____)`.

   From here, draw the next two levels of a proof tree that confirms your answer. Below each bottom node in your diagram, give the final minimax score from there.



10. **[4 marks]** Modify rock-paper-scissors so that Rose's payoff matrix is shown below. The game is 0-sum: Colin's payoff is the negative of Rose's. E.g. if Rose plays paper and Colin plays scissor then Rose loses 2 points and Colin gains 2 points. (A) What is Rose's expected payoff if she plays the mixed strategy (r 1/3, p 1/2, s 1/6) and Colin plays the pure strategy rock? (B) Repeat (A) when Colin plays paper. (C) Repeat (A) when Colin plays scissors. (D) The minimum of your answers to (A),(B),(C) is Rose's guaranteed expected payoff: she will alway win at least this amount. Give a mixed strategy for Rose that maximizes her guaranteed expected payoff.

```
   rock paper scissors      (A) _____
r    0   -1     1           (B) _____
p    1    0    -2           (C) _____
s   -1    2     0           (D)       (r ____, p ____, s ____)
```