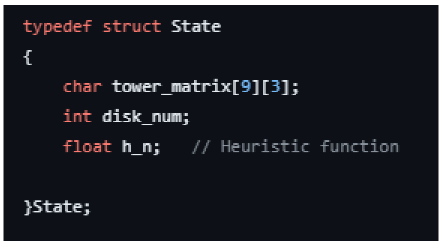
Definition of the Problem

1. State



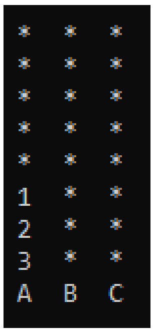
```
typedef struct State
{
    char tower_matrix[9][3];
    int disk_num;
    float h_n;    // Heuristic function

}State;
```

The asterisk (*) in the tower matrix represents spaces.

The letters (A,B,C) at the bottom of the tower matrix represent which tower we are in.

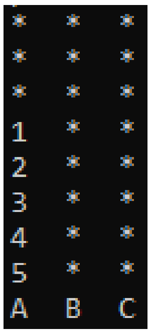
2. Initial State

a) Three Disks



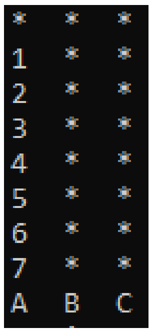
```
*  *  *
*  *  *
*  *  *
*  *  *
*  *  *
1  *  *
2  *  *
3  *  *
A  B  C
```

b) Five Disks



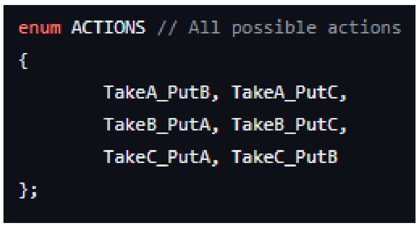
```
*  *  *
*  *  *
*  *  *
1  *  *
2  *  *
3  *  *
4  *  *
5  *  *
A  B  C
```

c) Seven Disks



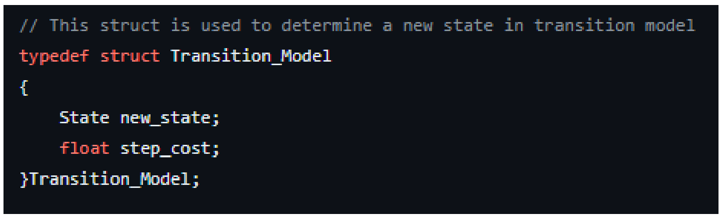
```
*  *  *
1  *  *
2  *  *
3  *  *
4  *  *
5  *  *
6  *  *
7  *  *
A  B  C
```

3. Actions



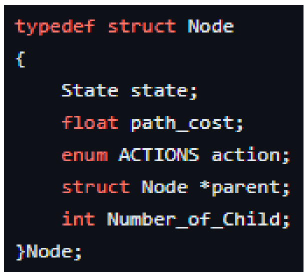
```
enum ACTIONS // All possible actions
{
        TakeA_PutB, TakeA_PutC,
        TakeB_PutA, TakeB_PutC,
        TakeC_PutA, TakeC_PutB
};
```

4. Transition Model



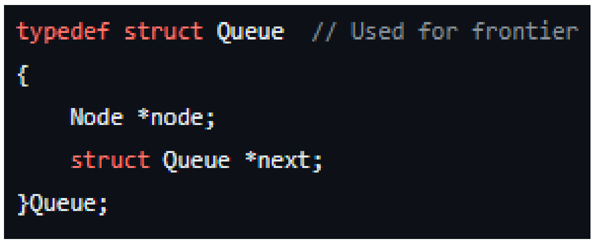
```
// This struct is used to determine a new state in transition model
typedef struct Transition_Model
{
    State new_state;
    float step_cost;
}Transition_Model;
```

5. Node



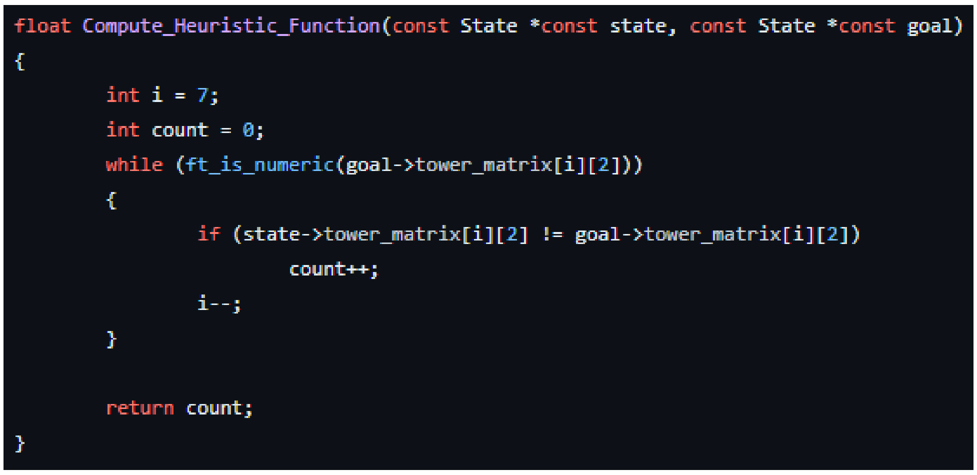
```
typedef struct Node
{
    State state;
    float path_cost;
    enum ACTIONS action;
    struct Node *parent;
    int Number_of_Child;
}Node;
```

6. Queue



```
typedef struct Queue  // Used for frontier
{
    Node *node;
    struct Queue *next;
}Queue;
```

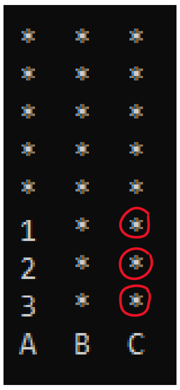
Definition of Heuristic Function



```
float Compute_Heuristic_Function(const State *const state, const State *const goal)
{
	int i = 7;
	int count = 0;
	while (ft_is_numeric(goal->tower_matrix[i][2]))
	{
		if (state->tower_matrix[i][2] != goal->tower_matrix[i][2])
			count++;
		i--;
	}

	return count;
}
```

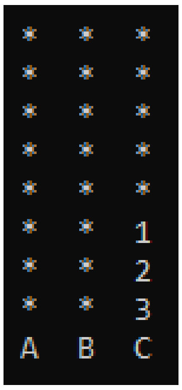
Returns the number of disks in the current state that differ in location from the disks in the goal state.

Example 1: In this example, our function will return the number 3.



```
*  *  *
*  *  *
*  *  *
*  *  *
*  *  *
1  *  *
2  *  *
3  *  *
A  B  C
```

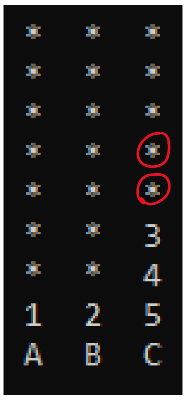
Current State



```
*  *  *
*  *  *
*  *  *
*  *  *
*  *  *
*  *  1
*  *  2
*  *  3
A  B  C
```

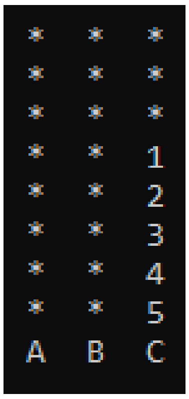
Goal State

Example 2: In this example, our function will return the number 2.



```
*  *  *
*  *  *
*  *  *
*  *  *
*  *  *
*  *  3
*  *  4
1  2  5
A  B  C
```

Current State

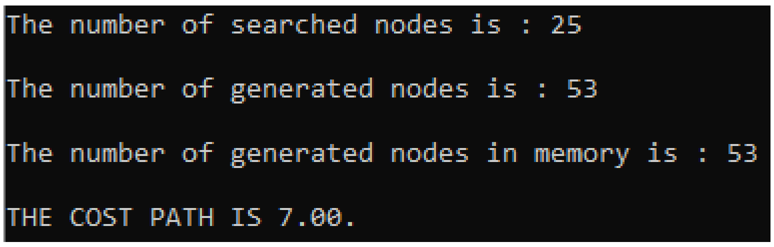


```
*  *  *
*  *  *
*  *  *
*  *  1
*  *  2
*  *  3
*  *  4
*  *  5
A  B  C
```

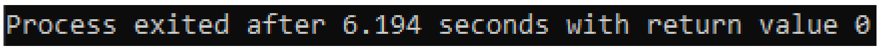
Goal State

Result of Several Simulations

a) **For three disks using Breast-First Search**

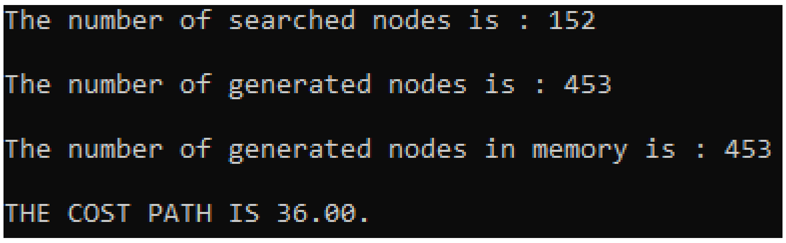


```
The number of searched nodes is : 25

The number of generated nodes is : 53

The number of generated nodes in memory is : 53

THE COST PATH IS 7.00.
```

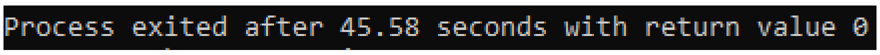


```
Process exited after 6.194 seconds with return value 0
```

b) **For five disks using A* Search**

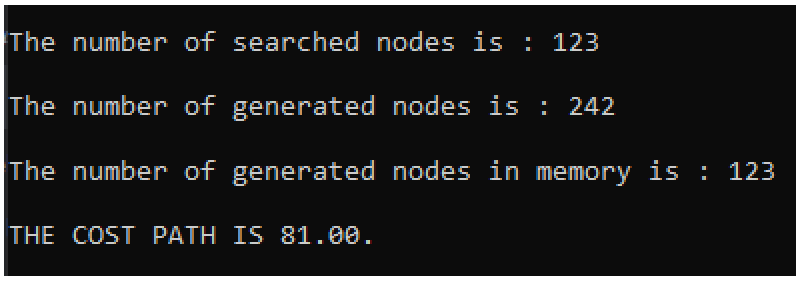


```
The number of searched nodes is : 152

The number of generated nodes is : 453

The number of generated nodes in memory is : 453

THE COST PATH IS 36.00.
```

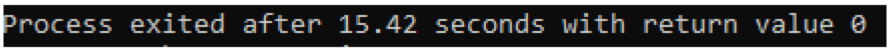


```
Process exited after 45.58 seconds with return value 0
```

c) **For five disks and maximum level is 500 using Depth-Limited Search**

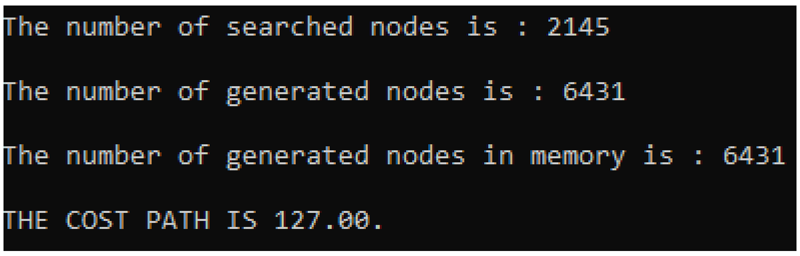


```
The number of searched nodes is : 123

The number of generated nodes is : 242

The number of generated nodes in memory is : 123

THE COST PATH IS 81.00.
```

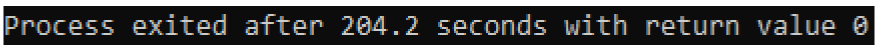


```
Process exited after 15.42 seconds with return value 0
```

d) **For seven disks using Uniform-Cost Search**



```
The number of searched nodes is : 2145

The number of generated nodes is : 6431

The number of generated nodes in memory is : 6431

THE COST PATH IS 127.00.
```



```
Process exited after 204.2 seconds with return value 0
```