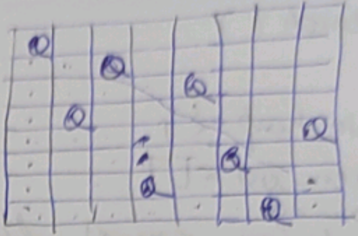
LAB (6):



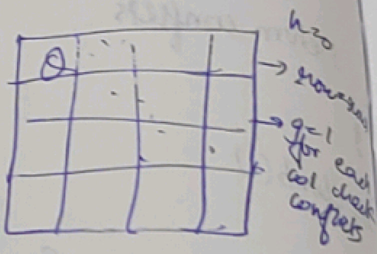
Hill climbing search for 8-queens:

```
conflicts()
{
  conflicts = 0
  for i ← 0 to 7
    for j ← i+1 to 7
      if (board[i] == board[j])....// checks if 2 queens can
        conflicts += 1              attack each other or not
                                    if yes, increment conflicks.

  return conflicts

}

hill-climb()
{
  randomly assign 8 queens in each row of the board.
  con = conflicts (board)
  ~~best-move = current-state)~~
  row = 0
  while (row <= rows)
      c = conflicts (board)
      if (current position of the queen is in conflicting position)
              cols ++

      else
        go to the next row and ~~conflicts~~ min (c, con)
      row++
  if (conflicts == 0)
      printf ("solution found! print the positions of the queen on board")

  else
      printf ("No solution found!")

}
```

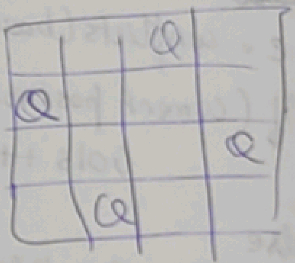
A* search for 8 queens:

```
conflicts()
{ conflicts = 0
  for i ← 0 to 7
    for j ← i+1 to 7
      check if 2 queens are in conflicting position
              conflicts ++
} return conflicts
```

```
A* search ( )
{
  min_heap = { }
  open_list = [ ]
  row = 0
  while (row < rows)  // check for each row
  {
    n = conflicts (board)
    if (conflicts == 0 && row == rows)
        printf("solution found")
        return
    min_heap.push (n)
    if (openlist [row] < min_heap.peek ( ))
        open_list.append (n)
  }
  return open_list
}
```



Proceed



~~Output~~

Code:

A* search:

```
def heuristic(board):
    conflicts =0
    for i in range(len(board)):
        for j in range(i+1, len(board)):
            if (board[i] == board[j] or abs(board[i]-board[j]) == i-j:
```

Output:

Solution board (column positions for each row): [0, 4, 7, 5, 2, 6, 1, 3]
Solution board (column positions for each row): [5, 3, 1, 7, 4, 6, 0, 2]