Day 1 (8 PUZZLE PROBLEM) :

```
class Solution:
  def solve(self, board):
    dict = {}
    flatten = []
    for i in range(len(board)):
      flatten += board[i]
    flatten = tuple(flatten)

    dict[flatten] = 0

    if flatten == (0, 1, 2, 3, 4, 5, 6, 7, 8):
      return 0

    return self.get_paths(dict)

  def get_paths(self, dict):
    cnt = 0
    while True:
      current_nodes = [x for x in dict if dict[x] == cnt]
      if len(current_nodes) == 0:
        return -1

      for node in current_nodes:
        next_moves = self.find_next(node)
        for move in next_moves:
          if move not in dict:
            dict[move] = cnt + 1
          if move == (0, 1, 2, 3, 4, 5, 6, 7, 8):
```

```
                return cnt + 1
            cnt += 1

    def find_next(self, node):
        moves = {
            0: [1, 3],
            1: [0, 2, 4],
            2: [1, 5],
            3: [0, 4, 6],
            4: [1, 3, 5, 7],
            5: [2, 4, 8],
            6: [3, 7],
            7: [4, 6, 8],
            8: [5, 7],
        }

        results = []
        pos_0 = node.index(0)
        for move in moves[pos_0]:
            new_node = list(node)
            new_node[move], new_node[pos_0] = new_node[pos_0], new_node[move]
            results.append(tuple(new_node))

        return results
ob = Solution()
matrix = [
    [3, 1, 2],
    [4, 7, 5],
    [6, 8, 0]
]
print(ob.solve(matrix))
```

DAY 1 (8 QUEEN) :

```
# Taking number of queens as input from user
print ("Enter the number of queens")
N = int(input())
# here we create a chessboard
# NxN matrix with all elements set to 0
board = [[0]*N for _ in range(N)]
def attack(i, j):
    #checking vertically and horizontally
    for k in range(0,N):
        if board[i][k]==1 or board[k][j]==1:
            return True
    #checking diagonally
    for k in range(0,N):
        for l in range(0,N):
            if (k+l==i+j) or (k-l==i-j):
                if board[k][l]==1:
                    return True
    return False
def N_queens(n):
    if n==0:
        return True
    for i in range(0,N):
        for j in range(0,N):
            if (not(attack(i,j))) and (board[i][j]!=1):
                board[i][j] = 1
                if N_queens(n-1)==True:
                    return True
                board[i][j] = 0
```

```
    return False
N_queens(N)
for i in board:
    print (i)
```

DAY 1 (WATER JUG) :

```
def pour(jug1, jug2):
    max1, max2, fill = 5, 7, 4  #Change maximum capacity and final capacity
    print("%d\t%d" % (jug1, jug2))
    if jug2 is fill:
        return
    elif jug2 is max2:
        pour(0, jug1)
    elif jug1 != 0 and jug2 is 0:
        pour(0, jug1)
    elif jug1 is fill:
        pour(jug1, 0)
    elif jug1 < max1:
        pour(max1, jug2)
    elif jug1 < (max2-jug2):
        pour(0, (jug1+jug2))
    else:
        pour(jug1-(max2-jug2), (max2-jug2)+jug2)

print("JUG1\tJUG2")
pour(0, 0)
```

DAY 1 (MISSONARIES – CANNIBAL PROBLEM) :

```
print("\n GAME STARTS ")

print("\n Now the task is to move all 3 cannibals and 3 missionaries from left to right  ")

print("\n the boat can carry 2 people at a time ")

print("\n If the cannibal value is greater than missionaries then the cannibal eat the massionaries ")

print("\n the boat cannot move other side without a people ")


lm = 3

lc = 3

rM = 0

rC = 0

userM = 0

userC = 0

k=0


print("\n C C C M M M |.....|         \n")


try:
   while(True):
      while(True):
         print("Left side -> right side of the river ")
         uM = int(input("enter the number of missionaries travel -> : "))
         uC = int(input("enter the nuumber of cannibals travel -> : "))
         if((uM==0) and (uC==0)):
            print("Empty travel is not possible ")
            print("re-emter the value : ")


         elif(((uM+uC)<= 2 )and((lm-uM)>=0) and ((lc-uC)>=0)):
            break
```

```
        else:
            print("Wrong input re enter the number : ")

    lm = (lm-uM)
    lc = (lc-uC)
    rM += uM
    rC += uC

    print("\n")
    for i in range(0,lm):
        print("M ",end=" ")
    for i in range(0,lc):
        print("C ",end=" ")
    print(" | -->  | ",end=" ")
    for i in range(0,rM):
        print("M ",end=" ")
    for i in range(0,rC):
        print("C ",end=" ")
    print("\n ")
    k +=1

    if(((lc==3) and lm==1)) or ((lc==3)and ((lm==2 ) and (lm==1)) or ((rC==3 )and (rM==1)) or
((rC==3)and (rM==2 )or (rC==2) and (rM==1))):
        print("cannibals eat missinories : \n you lost the game ")
        break
    if((rM+rC)==6):
        print("\n ypu won the game , \n congrats" )
        print("total attempts :")
        print(k)
        break
```

```
while(True):
   print("right side -> left side river travel ")
   userM=int(input("enter the number of missionaries : "))
   userC= int(input("enter the number of cannibals : "))

   if((userM==0)and(userC==0)):
      print("Empty travel not possible ")
      print("re enter the number : ")

   elif(((userM+userC)<=2 )and( (rM-userM)>=0) and((rC-userC)>=0)):
      break

   else:
      print("wron input re enter the number : ")

lm +=userM
lc +=userC
rM -=userM
rC -= userC
k+=1
print("\n")

for i in range(0,lm):
   print("M ",end=" ")
for i in range(0,lc):
   print("C ",end=" ")
print("| <-- | ",end=" ")
for i in range(0,rM):
   print("M ",end=" ")
for i in range(0,rC):
   print("C ",end=" ")
```

```
        print("\n")


        if (((lc == 3) and lm == 1)) or ((lc == 3) and ((lm == 2) and (lm == 1)) or ((rC == 3) and (rM == 1))
or ((rC == 3) and (rM == 2) or (rC == 2) and (rM == 1))):
            print("cannibals eat missinories : \n you lost the game ")
            break


except EOFError as e:
    print("\n invalid input ")
```

DAY 1 (CRIPT – ARITHAMITIC PROBLEM) :

```
public class SimpleSolver {
  static int eval(String q) {
    int val = 0;
    java.util.StringTokenizer st = new java.util.StringTokenizer(q, "*/+-", true);
    while (st.hasMoreTokens()) {
      String next = st.nextToken().trim();
      if (next.equals("+")) {
        val += Integer.parseInt(st.nextToken().trim());
      } else if (next.equals("-")) {
        val -= Integer.parseInt(st.nextToken().trim());
      } else if (next.equals("*")) {
        val *= Integer.parseInt(st.nextToken().trim());
      } else if (next.equals("/")) {
        val /= Integer.parseInt(st.nextToken().trim());
      } else {
        val = Integer.parseInt(next);
      }
    }
    return val;
```

```
}
static String solve(String q) {
  char c = 0;
  for (int i = 0; i < q.length(); ++i) {
    if (Character.isAlphabetic(q.charAt(i))) {
      c = q.charAt(i);
      break;
    }
  }
  if (c == 0) {
    String[] ops = q.split("==");
    int o1 = eval(ops[0]), o2 = eval(ops[1]);
    if (o1 == o2) return q;
    else return "";
  } else {
    char[] dset = new char[10];
    for (int i = 0; i < q.length(); ++i)
      if (Character.isDigit(q.charAt(i)))
        dset[q.charAt(i)-'0'] = 1;
    for (int i = 0; i < 10; ++i) {
      if (dset[i] == 0) {
        String r = solve(q.replaceAll(String.valueOf(c),
            String.valueOf(i)));
        if (!r.isEmpty()) return r;
      }
    }
  }
  return "";
}
public static void main(String[] args) {
  String query = "ABCDE * A == EEEEEE";
```

```
    System.out.println(solve(query));
  }
}
```

DAY 1(VACCUM CLEANER PROBLEM) :

```
import random

def display(room):
    print(room)

room = [
    [1, 1, 1, 1],
    [1, 1, 1, 1],
    [1, 1, 1, 1],
    [1, 1, 1, 1],
]
print("All the rooom are dirty")
display(room)

x =0
y= 0

while x < 4:
    while y < 4:
        room[x][y] = random.choice([0,1])
        y+=1
    x+=1
    y=0

print("Before cleaning the room I detect all of these random dirts")
```

```
display(room)
x =0
y= 0
z=0
while x < 4:
   while y < 4:
      if room[x][y] == 1:
         print("Vaccum in this location now,",x, y)
         room[x][y] = 0
         print("cleaned", x, y)
         z+=1
      y+=1
   x+=1
   y=0
pro= (100-((z/16)*100))
print("Room is clean now, Thanks for using : 3710933")
display(room)
print('performance=',pro,'%')
```