Create a python program to implement Hamiltonian circuit problem using Backtracking.

For example:

Result
Solution Exists: Following is one Hamiltonian Cycle 0 1 2 4 3 0

Answer: (penalty regime: 0 %)

Reset answer

```
class Graph():
    def __init__(self, vertices):
        self.graph = [[0 for column in range(vertices)]
                              for row in range(vertices)]
        self.V = vertices
    def isSafe(self, v, pos, path):
        if self.graph[ path[pos-1] ][v] == 0:
            return False
        for vertex in path:
            if vertex == v:
                return False

        return True
    def hamCycleUtil(self, path, pos):
        ######################Add your code here###################################
        #Start here
        if pos == self.V:
            if self.graph[ path[pos-1] ][ path[0] ] == 1:
                return True
            else:
                return False
```

	Expected	Got	
	Solution Exists: Following is one Hamiltonian Cycle 0 1 2 4 3 0	Solution Exists: Following is one Hamiltonian Cycle 0 1 2 4 3 0	

Passed all tests!

Correct

Marks for this submission: 20.00/20.00.

Question **2**

Correct

Mark 20.00 out of 20.00

Flag question

Write a python program using nested loop to find the prime numbers between 2 to 100.

For example:

Result
2 is prime 3 is prime 5 is prime 7 is prime 11 is prime 13 is prime 17 is prime 19 is prime 23 is prime 29 is prime 31 is prime 37 is prime 41 is prime 43 is prime 47 is prime 53 is prime 59 is prime 61 is prime 67 is prime 71 is prime 73 is prime 79 is prime 83 is prime 89 is prime 97 is prime Good bye!

Answer: (penalty regime: 0 %)

```
for num in range(2, 101):
    is_prime = True
    # Check for divisibility using a nested loop
```

```
    for i in range(2, int(num ** 0.5) + 1):
        if num % i == 0:
            is_prime = False
            break
    # If the number is prime, print it
    if is_prime:
        print(f"{num}  is prime")
print("Good bye!")

```

	Expected	Got	
	2 is prime 3 is prime 5 is prime 7 is prime 11 is prime 13 is prime 17 is prime 19 is prime 23 is prime 29 is prime 31 is prime 37 is prime 41 is prime 43 is prime 47 is prime 53 is prime 59 is prime 61 is prime 67 is prime 71 is prime 73 is prime 79 is prime 83 is prime 89 is prime 97 is prime Good bye!	2 is prime 3 is prime 5 is prime 7 is prime 11 is prime 13 is prime 17 is prime 19 is prime 23 is prime 29 is prime 31 is prime 37 is prime 41 is prime 43 is prime 47 is prime 53 is prime 59 is prime 61 is prime 67 is prime 71 is prime 73 is prime 79 is prime 83 is prime 89 is prime 97 is prime Good bye!	

Passed all tests!

Correct

Marks for this submission: 20.00/20.00.

Question **3**

Correct

Mark 20.00 out of 20.00

Flag question

Write a Python program for Bad Character Heuristic of Boyer Moore String Matching Algorithm

For example:

Input	Result
ABAAAABCD ABC	Pattern occur at shift = 5

Answer: (penalty regime: 0 %)

Reset answer

```
NO_OF_CHARS = 256
def badCharHeuristic(string, size):
    ############ Add your Code Here ################
    badChar = [-1] * NO_OF_CHARS
    for i in range(size):
        badChar[ord(string[i])] = i
    return badChar
def search(txt, pat):
    m = len(pat)
    n = len(txt)
    badChar = badCharHeuristic(pat, m)
    s = 0
    while(s <= n-m):
        j = m-1
        while j>=0 and pat[j] == txt[s+j]:
            j -= 1
        if j<0:
```

```
            print("Pattern occur at shift = {}".format(s))
            s += (m-badChar[ord(txt[s+m])] if s+m<n else 1)
        else:
            s += max(1, j-badChar[ord(txt[s+j])])
def main():
```

	Input	Expected	Got	
	ABAAAABCD ABC	Pattern occur at shift = 5	Pattern occur at shift = 5	

Passed all tests!

Correct

Marks for this submission: 20.00/20.00.

Question **4**

Correct

Mark 20.00 out of 20.00

Flag question

Write a python program to implement Boyer Moore Algorithm with Good Suffix heuristic to find pattern in given text string.

For example:

Input	Result
ABAAABAACD ABA	pattern occurs at shift = 0 pattern occurs at shift = 4

Answer: (penalty regime: 0 %)

Reset answer

```
def preprocess_strong_suffix(shift, bpos, pat, m):
    ###############  Add your Code here ####################
    i = m
    j = m + 1
    bpos[i] = j
    while i > 0:
        while j <= m and pat[i - 1] != pat[j - 1]:
            if shift[j] == 0:
                shift[j] = j - i
            j = bpos[j]
        i -= 1
        j -= 1
        bpos[i] = j

def preprocess_case2(shift, bpos, pat, m):
    j = bpos[0]
    for i in range(m + 1):
        if shift[i] == 0:
            shift[i] = j
        if i == j:
            j = bpos[j]
def search(text, pat):
```

	Input	Expected	Got	
	ABAAABAACD ABA	pattern occurs at shift = 0 pattern occurs at shift = 4	pattern occurs at shift = 0 pattern occurs at shift = 4	
	SaveethaEngineering Saveetha veetha	pattern occurs at shift = 2 pattern occurs at shift = 22	pattern occurs at shift = 2 pattern occurs at shift = 22	

Passed all tests!

Correct

Marks for this submission: 20.00/20.00.

Question **5**

Correct

Mark 20.00 out of 20.00

Flag question

Write a python program to implement knight tour problem

For example:

Input	Result
5 5	[1, 12, 25, 18, 3] [22, 17, 2, 13, 24] [11, 8, 23, 4, 19] [16, 21, 6, 9, 14]

Input	Result
	[7, 10, 15, 20, 5] [(0, 0), (1, 2), (0, 4), (2, 3), (4, 4), (3, 2), (4, 0), (2, 1), (3, 3), (4, 1), (2, 0), (0, 1), (1, 3), (3, 4), (4, 2) Done!

Answer: (penalty regime: 0 %)

Reset answer

```
import sys
class KnightsTour:
    def __init__(self, width, height):
        self.w = width
        self.h = height
        self.board = []
        self.generate_board()

    def generate_board(self):
        for i in range(self.h):
            self.board.append([0]*self.w)

    def print_board(self):

        for elem in self.board:
            print (elem)

    def generate_legal_moves(self, cur_pos):
        possible_pos = []
        move_offsets = [(1, 2), (1, -2), (-1, 2), (-1, -2),
                        (2, 1), (2, -1), (-2, 1), (-2, -1)]
        ############### Add your code here ################################
```

	Input	Expected
	5 5	[1, 12, 25, 18, 3] [22, 17, 2, 13, 24] [11, 8, 23, 4, 19] [16, 21, 6, 9, 14] [7, 10, 15, 20, 5] [(0, 0), (1, 2), (0, 4), (2, 3), (4, 4), (3, 2), (4, 0), (2, 1), (3, 3), (4, 1), (2, 0), (0, 1), (1, 3), (3, 4), Done!
	6 6	[1, 32, 9, 18, 3, 34] [10, 19, 2, 33, 26, 17] [31, 8, 25, 16, 35, 4] [20, 11, 36, 27, 24, 15] [7, 30, 13, 22, 5, 28] [12, 21, 6, 29, 14, 23] [(0, 0), (1, 2), (0, 4), (2, 5), (4, 4), (5, 2), (4, 0), (2, 1), (0, 2), (1, 0), (3, 1), (5, 0), (4, 2), (5, 4), Done!

Passed all tests!

Correct

Marks for this submission: 20.00/20.00.