# Bahria University,

## Karachi Campus



LAB EXPERIMENT NO.

**\_07\_**

LIST OF TASKS

|  |  |
| --- | --- |
| **TASK NO** | **OBJECTIVE** |
| 01 | Using a genetic Algorithm Create a GUI based Guess a password given the number of correct letters in the guess. Build a mutation Engine |
|  |  |
|  |  |
|  |  |
|  |  |

Submitted On:

Date: 02/04/2024

**Task No 01:** Using a genetic Algorithm Create a GUI based Guess a password given the number of correct letters in the guess. Build a mutation Engine

**Solution:**

import random

import string

def generate\_password(length):

    return ''.join(random.choice(string.ascii\_letters + string.digits) for \_ in range(length))

def calculate\_fitness(guess, target\_password):

    return sum(1 for expected, actual in zip(target\_password, guess) if expected == actual)

def genetic\_algorithm(target\_password, population\_size, mutation\_rate):

    population = [generate\_password(len(target\_password)) for \_ in range(population\_size)]

    while True:

        fitness\_scores = [(guess, calculate\_fitness(guess, target\_password)) for guess in population]

        best\_guess, best\_fitness = max(fitness\_scores, key=lambda x: x[1])

        if best\_fitness == len(target\_password):

            return best\_guess

        parents = [guess for guess, fitness in fitness\_scores if fitness > 0]

        if not parents:

            parents = population

        new\_generation = []

        while len(new\_generation) < population\_size:

            parent1, parent2 = random.sample(parents, 2)

            crossover\_point = random.randint(1, len(target\_password) - 1)

            child = parent1[:crossover\_point] + parent2[crossover\_point:]

            new\_generation.append(child)

        for i in range(len(new\_generation)):

            if random.random() < mutation\_rate:

                index\_to\_mutate = random.randint(0, len(target\_password) - 1)

                new\_generation[i] = new\_generation[i][:index\_to\_mutate] + random.choice(string.ascii\_letters + string.digits) + new\_generation[i][index\_to\_mutate + 1:]

        population = new\_generation

target\_password = input("Enter the target password: ")

num\_correct\_letters = int(input("Enter the number of correct letters in the guess: "))

best\_guess = genetic\_algorithm(target\_password, 100, 0.1)

print("Best Guess:", best\_guess)

**A screenshot of a computer

Description automatically generated**

**Output:**