# Bahria University,

# A logo with text on it Description automatically generatedKARACHI 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: 25/2/2023

**TASK 1:**Using a genetic Algorithm Create a GUI based Guess a password given the number of correct letters in the guess. Build a mutation Engine

import random

import tkinter as tk

from tkinter import scrolledtext

class GeneticAlgorithm:

def \_\_init\_\_(self, target, population\_size=100, mutation\_rate=0.01):

self.target = target

self.population\_size = population\_size

self.mutation\_rate = mutation\_rate

self.population = self.initialize\_population()

self.generation = 0

def initialize\_population(self):

population = []

for \_ in range(self.population\_size):

individual = ''.join(random.choice('abcdefghijklmnopqrstuvwxyz') for \_ in range(len(self.target)))

population.append(individual)

return population

def fitness(self, individual):

return sum(1 for i, j in zip(individual, self.target) if i == j)

def selection(self):

sorted\_population = sorted(self.population, key=self.fitness, reverse=True)

return sorted\_population[:2]

def crossover(self, parent1, parent2):

child = ''

for i in range(len(self.target)):

if random.random() < 0.5:

child += parent1[i]

else:

child += parent2[i]

return child

def mutate(self, individual):

mutated = ''

for char in individual:

if random.random() < self.mutation\_rate:

mutated += random.choice('abcdefghijklmnopqrstuvwxyz')

else:

mutated += char

return mutated

def evolve(self):

new\_generation = []

parents = self.selection()

best\_guess = max(self.population, key=self.fitness)

best\_fitness = self.fitness(best\_guess)

if best\_guess == self.target:

new\_generation = [self.target] \* self.population\_size

else:

for \_ in range(self.population\_size):

child = self.crossover(parents[0], parents[1])

child = self.mutate(child)

new\_generation.append(child)

self.population = new\_generation

self.generation += 1

return best\_guess, best\_fitness, best\_guess == self.target

def run\_genetic\_algorithm():

target = password\_entry.get()

ga = GeneticAlgorithm(target=target, population\_size=100, mutation\_rate=0.01)

match\_found = False

all\_guesses.delete('1.0', tk.END) # Clear previous guesses

while not match\_found:

best\_guess, best\_fitness, match\_found = ga.evolve()

guesses\_display = ' '.join(ga.population)

all\_guesses.insert(tk.END, f"Generation {ga.generation}: {guesses\_display}\n")

result\_label.config(text=f"Best Guess: {best\_guess} - Generation: {ga.generation} - Correct Letters: {best\_fitness}")

root.update\_idletasks()

if match\_found:

result\_label.config(text=f"Password matched: {best\_guess} - Generation: {ga.generation} - Fitness: 0")

break

root = tk.Tk()

root.title("Password Guesser")

root.geometry("400x400")

root.configure(bg="#f0f0f0")

instruction\_label = tk.Label(root, text="Enter Password:", bg="#f0f0f0", font=("Helvetica", 12))

instruction\_label.pack(pady=10)

password\_entry = tk.Entry(root)

password\_entry.pack(pady=5)

start\_button = tk.Button(root, text="Start Guessing", command=run\_genetic\_algorithm, bg="#4CAF50", fg="white", font=("Helvetica", 12))

start\_button.pack(pady=10)

result\_label = tk.Label(root, text="Best Guess: - Generation: 0 - Correct Letters: 0", bg="#f0f0f0", font=("Helvetica", 10))

result\_label.pack(pady=5)

all\_guesses = scrolledtext.ScrolledText(root, height=10)

all\_guesses.pack(pady=10, padx=5, fill="both", expand=True)

root.mainloop()

