CS112 – Structured Programming
Assignment-1

Team members:

No.	Names	IDs
1.	Adham Wael Mansour	20210057
2.	Alaa Azazi Abd El-Hamid	20210071
3.	Esraa Ahmed Saad	20210062

Group task:
Application 0 – Population Application with openpyxl

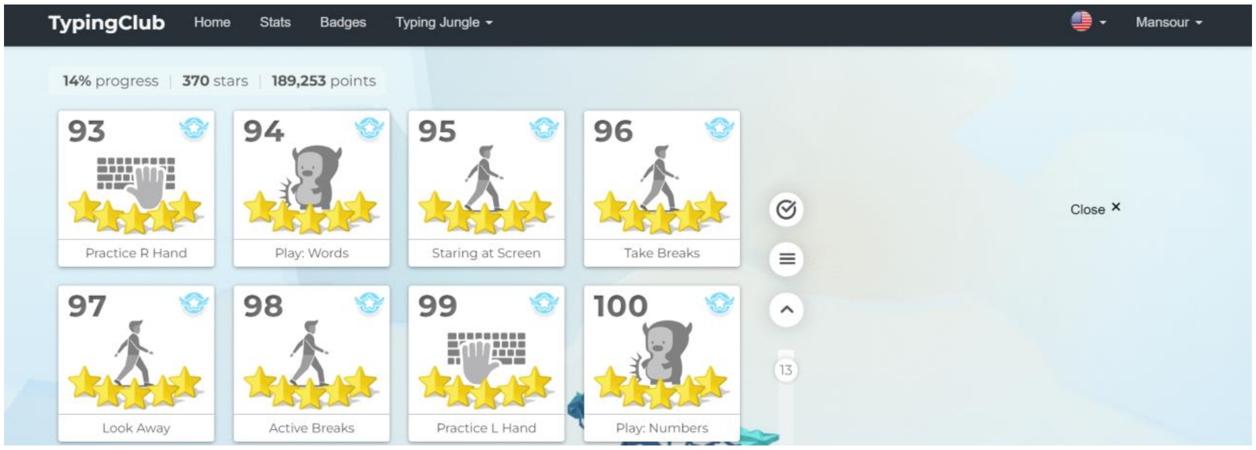
Individual tasks:
Adham (20210057):
- Task 2: Subtract a square game
- Bonus: Subtract a square game in C++

Alaa (20210071):
- Task 2: Connect 4 game

Esraa (20210062):
- Task 2: Tic-Tac-Toe with numbers

Task 1:

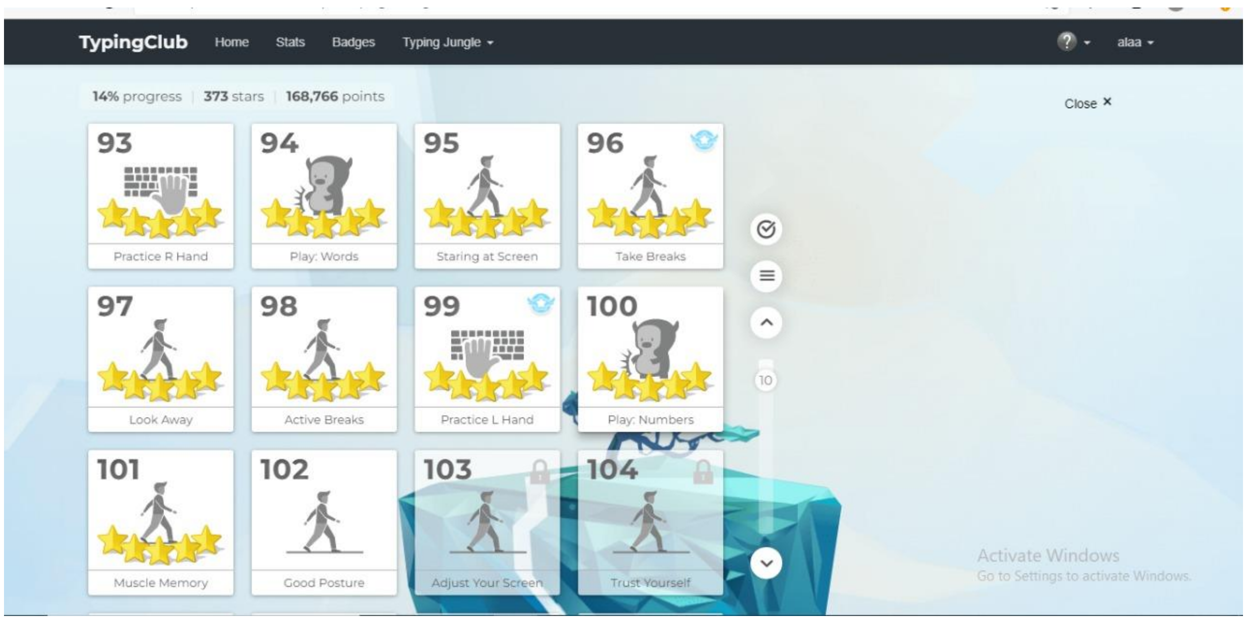
Adham Wael (20210057)



14% progress | 370 stars | 189,253 points

93 Practice R Hand	94 Play: Words	95 Staring at Screen	96 Take Breaks
97 Look Away	98 Active Breaks	99 Practice L Hand	100 Play: Numbers

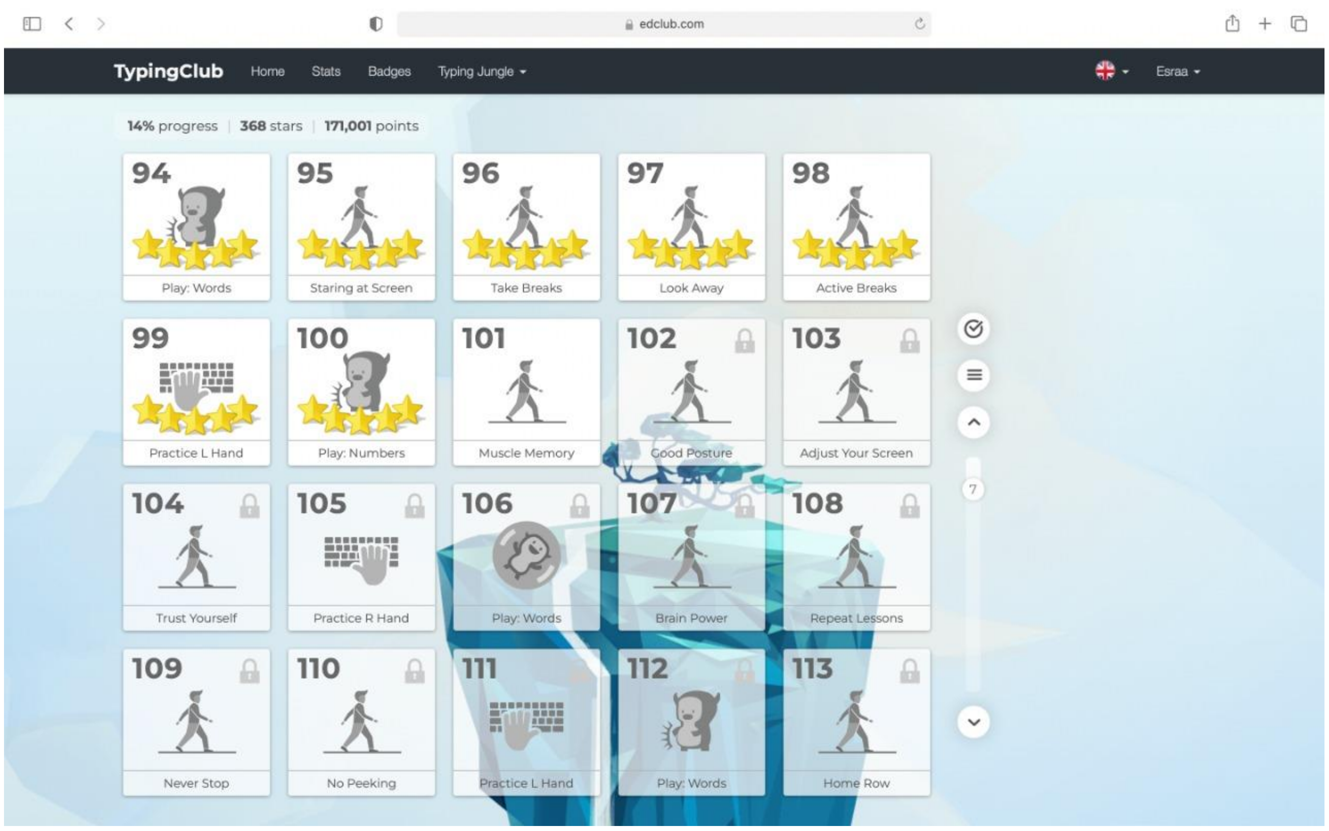
Alaa Azazi (20210071)



14% progress | 373 stars | 168,766 points

93 Practice R Hand	94 Play: Words	95 Staring at Screen	96 Take Breaks
97 Look Away	98 Active Breaks	99 Practice L Hand	100 Play: Numbers
101 Muscle Memory	102 Good Posture	103 Adjust Your Screen	104 Trust Yourself

Esraa Ahmed (20210062)



Task 2:
Adham Wael (20210057)
Subtract a square game:

Video:
https://drive.google.com/file/d/1G5WM3Du0ZW7qEwMCBMxrd9YpMqfv9pyQ/view?usp=sharing

Algorithm:

1. Choose a random number of tokens
2. Make a list to store the squared numbers to force the user to choose from them
3. Make a loop to store all the squared numbers until the number of tokens
4. Print the number of tokens
5. Make a for loop to take input from the players
6. Take input from first player
7. Check if the input is squared
8. Subtract the number from the total number of tokens
9. Check if the number of tokens equals 0 to announce the winner
10. Repeat the operation with player 2

Source Code:
```
# This is subtracting a square game
# Author : Adham mansour (20210057)
# Date : 24/2/2022
import math

no_of_tokens = 50  # just random number of tokens
squared_numbers = []


def generating_numbers():  # this should generate all the squared numbers from 0
to number of tokens
```

```
    i = 1
    while i ** 2 <= no_of_tokens:
        squared_numbers.append(i ** 2)
        i += 1


def choosing_numbers():  # this is where the real game starts
    global no_of_tokens

    print("The number of tokens is ", no_of_tokens)

    for k in range(math.ceil(no_of_tokens / 2)):  # the math function is used
here to round up the fraction if the
        # number of tokens is odd
        number_of_player1 = int(input("Player 1: Please enter number: "))
        while number_of_player1 not in squared_numbers:  # this should check if
the number given is squared
            print("number doesn't exist")
            number_of_player1 = int(input("Please enter number: "))
        no_of_tokens -= number_of_player1
        if no_of_tokens == 0:  # this should check if first player won the game
if the list is empty
            print("Player 1 wins")
            break
        print("Remaining: ", no_of_tokens)
        number_of_player2 = int(input("Player 2: Please enter number: "))
        while number_of_player2 not in squared_numbers:
            print("number doesn't exist")
            number_of_player2 = int(input("Please enter number: "))
        no_of_tokens -= number_of_player2
        if no_of_tokens == 0:
            print("Player 2 wins")
            break
        print("Remaining: ", no_of_tokens)


def all_game():
    generating_numbers()
    choosing_numbers()


all_game()
```

Alaa Azazi (20210071)
Connect 4 game:

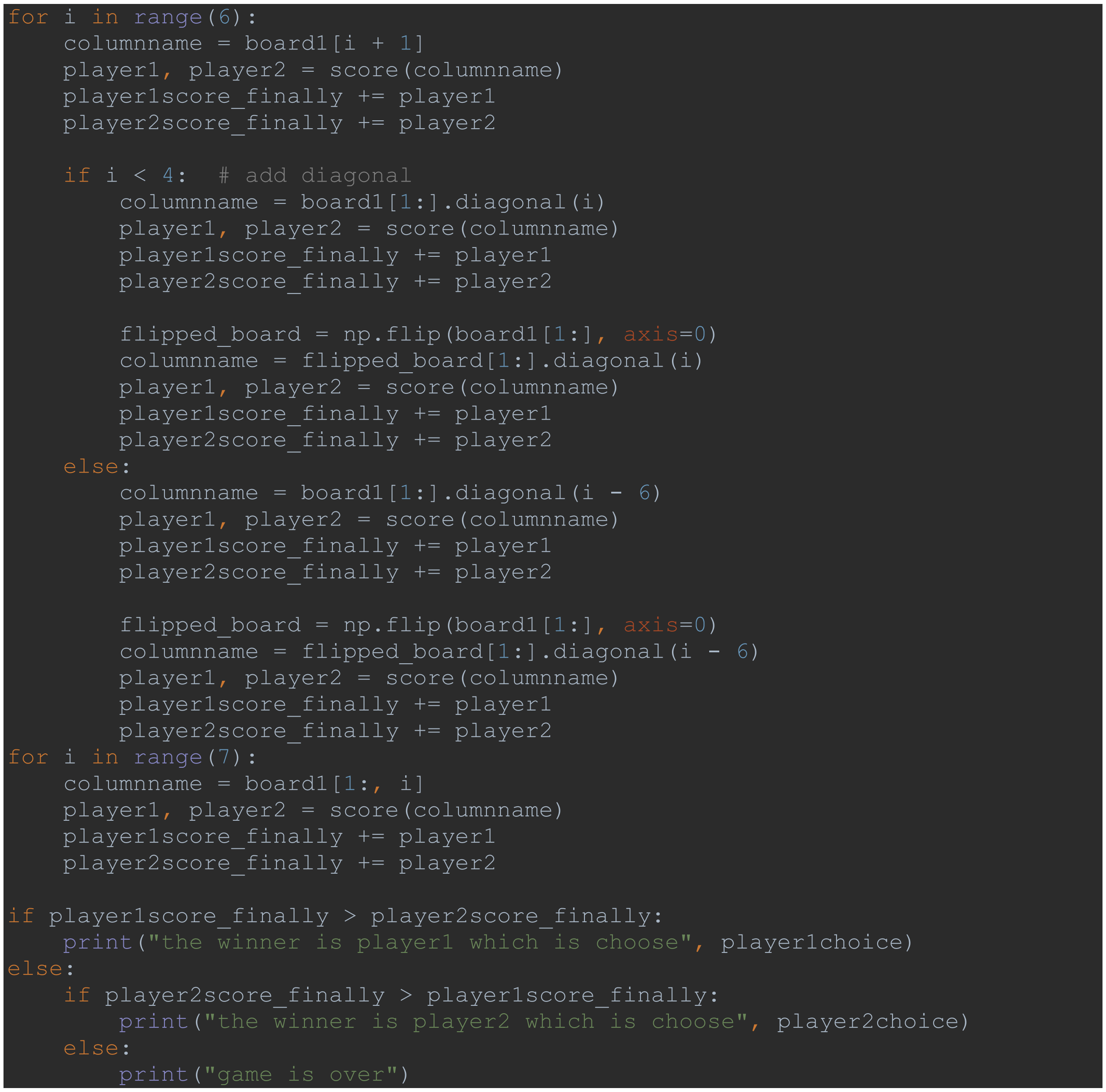
Video link:
https://drive.google.com/file/d/1EYOTLfz_YxHDMZ80PfmjZIL1IdCuq3Dk/view?usp=sharing

Algorithm:

1. Make a board is 7*7
2. Each column has a character
3. make a user input the mark
4. Make a players enter the column
5. Put your mark in the last column
6. calculate the score
7. Check the winner

Source code:

```
#connect 4 game

import numpy as np

# creat a board
board1 = np.zeros((7, 7), dtype=str)
import string

board1[0] = list(string.ascii_uppercase)[0:7]
board1 = np.where(board1 == '', ' ', board1)
# ask player1 about input
player1choice = input('please enter (x , o):').lower()
if player1choice == 'x':
    player2choice = 'o'
else:
    player2choice = 'x'
# put yuor mark in the last column
column_dict = {'A': 0, 'B': 1, 'C': 2, 'D': 3, 'E': 4, 'F': 5, 'G': 6}


def player1turn():
    columnname = input("frist player, please choose a column from A to G :
").upper()
    columnindx = column_dict[columnname]
    totalcells = board1[1:, columnindx]  # take the second of row and
columnindx
    empty = (totalcells == ' ').sum()
    if empty == 6:
        totalcells[-1] = player1choice
    else:
        totalcells[empty - 7] = player1choice
    print(board1)


def player2turn():
    columnname = input("second player, please choose a column from A to G:
").upper()
    columnindx = column_dict[columnname]
    totalcells = board1[1:, columnindx]  # take the second of row and
columnindx
    empty = (totalcells == ' ').sum()
    if empty == 6:
        totalcells[-1] = player2choice
    else:
        totalcells[empty - 7] = player2choice
    print(board1)


for i in range(21):
    player1turn()
    player2turn()
print("game over")


# calculation score

def score(columnname):
    xscore = 0
    oscore = 0
    for cell in columnname:
        if cell == "x":
            xscore += 1
    else:
        if xscore % 4 != 0:
            xscore = xscore - (xscore % 4)

    for cell in columnname:
        if cell == "o":
            oscore += 1
    else:
        if xscore % 4 != 0:
            oscore = oscore - (oscore % 4)

    if player1choice == 'X':
        player1score = xscore // 4
        player2score = oscore // 4
    else:
        player2score = xscore // 4
        player1score = oscore // 4
    return player1score, player2score


player1score_finally = 0
player2score_finally = 0
```



```python
for i in range(6):
    columnname = board1[i + 1]
    player1, player2 = score(columnname)
    player1score_finally += player1
    player2score_finally += player2

    if i < 4:  # add diagonal
        columnname = board1[1:].diagonal(i)
        player1, player2 = score(columnname)
        player1score_finally += player1
        player2score_finally += player2

        flipped_board = np.flip(board1[1:], axis=0)
        columnname = flipped_board[1:].diagonal(i)
        player1, player2 = score(columnname)
        player1score_finally += player1
        player2score_finally += player2
    else:
        columnname = board1[1:].diagonal(i - 6)
        player1, player2 = score(columnname)
        player1score_finally += player1
        player2score_finally += player2

        flipped_board = np.flip(board1[1:], axis=0)
        columnname = flipped_board[1:].diagonal(i - 6)
        player1, player2 = score(columnname)
        player1score_finally += player1
        player2score_finally += player2
for i in range(7):
    columnname = board1[1:, i]
    player1, player2 = score(columnname)
    player1score_finally += player1
    player2score_finally += player2

if player1score_finally > player2score_finally:
    print("the winner is player1 which is choose", player1choice)
else:
    if player2score_finally > player1score_finally:
        print("the winner is player2 which is choose", player2choice)
    else:
        print("game is over")
```

Esraa Ahmed (20210062)
Tic-Tac-Toe game:

Video link: https://moebh-my.sharepoint.com/:v:/g/personal/amira_t_korany5253_moe_bh/EV18hNIuMJ5KsKDTk118Y04Bf5vvAD7Rjuc8L1p1LmUrlw?e=UIzmyB

Algorithm:

1. Making a 3*3 board
2. Each player has some specific numbers
3. The first player is allowed to add odd numbers only and the second is even numbers
4. Each player select the number and its position in the board
5. How to check the winner?
6. How to get the sum of each row, every column, and every diameter
7. When the sum = 15 then we have a winner

Source code:

```
board = [0,1,2,
    3,4,5,
    6,7,8]
boardLog = [0, 0, 0,
       0, 0, 0,
       0, 0, 0]

player = '1'

def tic_tac_toe ():
   print ('|' ,board[0],'|',board[1] ,'|', board[2],'|')
   print ('--------------------')
   print ('|' ,board[3],'|',board[4] ,'|', board[5],'|')
   print ('--------------------')
   print ('|' ,board[6],'|',board[7] ,'|', board[8],'|')

def move(x1,x2):
   board[x2] = x1
   boardLog[x2] = 1
   tic_tac_toe()

def odd (x, x2):
   while  (x%2==0):
       x = int(input ('enter an odd number'))
   move (x ,x2)

def even (x ,x2) :
   while  (x%2!=0):
       x = int(input ('enter an even number'))
   move (x ,x2)

def winner():
   if (boardLog[0] + boardLog[1] + boardLog[2] == 3):
     if (board[0]+board [1]+board[2]==15):
         print ('you are the winner')
         return True
   if (boardLog[0] + boardLog[3] + boardLog[6] == 3):
     if (board[0]+board [3]+board[6]==15):
         print ('you are the winner')
         return True
   if (boardLog[1] + boardLog[4] + boardLog[7] == 3):
     if (board[1]+board [4]+board[7]==15):
         print ('you are the winner')
         return True
   if (boardLog[3] + boardLog[4] + boardLog[5] == 3):
     if (board[3]+board [4]+board[5]==15):
         print ('you are the winner')
         return True
   if (boardLog[2] + boardLog[5] + boardLog[8] == 3):
     if (board[2]+board [5]+board[8]==15):
         print ('you are the winner')
         return True
   if (boardLog[6] + boardLog[7] + boardLog[8] == 3):
     if (board[6]+board [7]+board[8]==15):
         print ('you are the winner')
         return True

   else: return False

def turn(s):
   print ('its '+ s +' turn')
   x = int (input ('enter the number: '))
   x1 = int (input ('enter the places number: '))
   if player == 'a':
       even(x, x1)
   else: odd(x, x1)

print('Tic Tac Toe')
print ('player 1 should enter odd numbers only'+' and player 2 should enter even numbers only')
print ('the player with the ood numbers start')
tic_tac_toe ()
while (True):
 turn(player)
 if winner():
    break
```

Task 3:

Application 0

Video link: https://drive.google.com/file/d/1Gtd4F9jJVye-yrPHawGZ2fP_2383792w/view?usp=sharing

Algorithm:

1. Define a function to take the country from user and open and load data from the excel file.
2. Define a function that display all the states and their population.
3. Make a loop inside the display content function to loop through all rows and display all the data.
4. Define a function to store the states and their population respectively to be used to find the maximum population later.
5. Define a function find max population to find the maximum population and match it with it's correspondent state.
6. Repeat the same function but find the minimum value instead.
7. Define a function to display options to the user and execute functions according to the user choice.
8. Make a try except statement to handle the FileNotFound error and display a message to the user instead.

Source code:

```python
# Project 0 to open excel file and read data from it

import openpyxl


def open_file():
    file_path = input("enter the file path: ")
    file = openpyxl.load_workbook(file_path)
    global sheet
    sheet = file.active
    global lastrow
    lastrow = sheet.max_row


def displaying_content(lastrow, sheet):
    for r in range(1, lastrow + 1):
        state = chr(65)  # this should read the date from the first column
        population = chr(66)  # this should read the date from the second column
        index_state = state + str(r)
        population_state = population + str(r)
        print(sheet[index_state].value, " ",
              sheet[population_state].value)  # this should print the state with
it's population


def appending_values(lastrow):
    global populations, states
    populations = []
    states = []
    for k in range(1, lastrow + 1):
        populations.append(sheet.cell(row=k, column=2).value)
        states.append(sheet.cell(row=k, column=1).value)


def find_max_population(lastrow, populations, states):  # this should find the
country with the greatest population
    # those are filled later to be used with the max() function

    max_population = max(populations)
    index_max_state = populations.index(max_population)

    print("The state with max population is:", states[index_max_state])
```

```
def find_min_population(lastrow, populations,states):
    # this should work like the find_max population function but with min()

    min_population = min(populations)
    index_min_state = populations.index(
        min_population)  # this should get the index of the state with the
highest population

    print("The state with min population is:", states[index_min_state])


def choices():
    open_file()
    print("Enter 1 to display population of each state: ")
    print("Enter 2 to display the state with the highest and lowest population:
")
    print("Enter 3 to exit")
    choice = int(input())

    if choice == 1:
        displaying_content(lastrow, sheet)
    elif choice == 2:
        appending_values(lastrow)
        find_max_population(lastrow, populations, states)
        find_min_population(lastrow, populations, states)
    else:
        quit()


try:
    choices()
except FileNotFoundError:
    print("Country doesn't exist")
```

Bonus tasks:

Adham Wael (20210057)

Task 5:
Subtract a square game in C++

Source code:

```
//This is subtracting a square game
// Created by Adham mansour(20210057) on 2/24/2022.

#include <iostream>
#include <list>
#include<bits/stdc++.h>

using namespace std;

int max_no_of_tokens = 50;
std::list<int> squared_numbers = {};

void generating_numbers(){ // this should generate the numbers and store them
in a list
    int i = 1;
    while (i * i <= max_no_of_tokens){
        squared_numbers.push_back(i * i);
        i++;
    }

}

void printing_the_list(){ // this should print the number in the list to be
used after every move
    for (int x : squared_numbers) {
        cout << x << ' ';
    }
    cout << endl;
}


void choosing_numbers(){ // This is where the real fun begins

    cout << "The number of tokens is: " << max_no_of_tokens << endl;

    int number_of_player_1, number_of_player_2;

    for(int k = 0; k <= squared_numbers.size(); k++) { //this is where the
real game starts
        cout << "Player 1 : Please enter the number: ";
        cin >> number_of_player_1;
        if (bool (std::find(squared_numbers.begin(), squared_numbers.end(),
number_of_player_1) !=
                          squared_numbers.end())) { // this should check if
the number entered exists in the list or not
            squared_numbers.remove(number_of_player_1);
            max_no_of_tokens-= number_of_player_1;
        }
        else {
            cout << "Number doesn't exist enter another number: ";
            cin >> number_of_player_1;
            squared_numbers.remove(number_of_player_1);
            max_no_of_tokens-= number_of_player_1;
        }
        if (max_no_of_tokens == 0) { // this should decide on the winner
            cout << "Player 1 wins" << endl;
            break;
        }

        cout << "The remaining tokens: " << max_no_of_tokens << endl;

        cout << "Player 2 : Please enter the number: "; //same procedure as
before but with player 2
        cin >> number_of_player_2;

        if (bool (std::find(squared_numbers.begin(), squared_numbers.end(),
number_of_player_2) !=
                          squared_numbers.end())) {
            squared_numbers.remove(number_of_player_2);
            max_no_of_tokens-= number_of_player_2;
        }
        else {
            cout << "Number doesn't exist enter another number: ";
            cin >> number_of_player_2;
            squared_numbers.remove(number_of_player_2);
            max_no_of_tokens-= number_of_player_1;
        }
        if (max_no_of_tokens == 0) {
            cout << "Player 2 wins" << endl;
            break;
        }
        cout << "The remaining tokens: " << max_no_of_tokens << endl;
    }
}

int main(){ // this should recall all the game functions
    generating_numbers();
    choosing_numbers();

}
```