Name: Muhammad Anas Baig

Enrollment No.: 01-134152-037

Section: BS(CS)-5A

ASSIGNMENT # 2

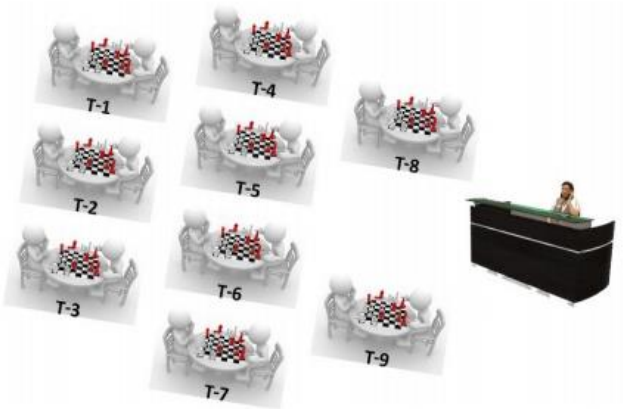
Visual Programming

BS(CS) – 5A

October 23, 2017

For Assignment 1, you're required to prepare Visual Application using Windows Form Application C^3 (Chess Challenge Consortium)

C Cube (C^3) is Chess Challenge Consortium where players reserve their tables for chess games with opponents or without opponents. In case of a player arrives without opponent, the manager on reception desk helps him to find the opponent. Game is played and results of the game is stored by the manager along with the information of players, table id and date time. There are total nine tables available at a time. In case all tables are busy, the player has to wait till the result of ongoing game is concluded. Your task is to design a system which will store the information of players, their reservations and results of the game they played with the opponent. Your system should have at least the following features:



1. System should allow user to register the information of player such as, Name CNIC. Only if player is not already registered.
2. In case of 2 players arrive to play, the table should be assigned immediately.
3. In case player single player is arrived, your system should reserve table for him, or should assigned already single player waiting for game.
4. In case 2 players arrive to play and table is not available then the reserved table of single player should be assigned to them for immediate start of game.
5. Result of the game should be store along with the table id, date and time. Result of game should be saved with player's statistics.
6. System should show the current status of all the tables.
7. System should be able to search a player's information by any mean along with total games he played and total wins/draw/lose.
8. System should store data in file(s).

Source Code:

Form1.cs Code:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.IO;
using System.Collections;

namespace Assignment2
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
            displayTable();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            Form2 f = new Form2();
            f.Show();
            displayTable();
        }

        private void button2_Click(object sender, EventArgs e)
        {
            Form3 f = new Form3();
            f.Show();
            displayTable();
        }

        private void button3_Click(object sender, EventArgs e)
        {

        }

        private void button3_Click_1(object sender, EventArgs e)
        {
            Form4 f = new Form4();
            f.Show();
            displayTable();
        }

        private void button4_Click(object sender, EventArgs e)
        {
            Form5 f = new Form5();
            f.Show();
            displayTable();
        }

        private void button5_Click(object sender, EventArgs e)
        {
            Form6 f = new Form6();
            f.Show();
            displayTable();
```

```
        }

        private void button6_Click(object sender, EventArgs e)
        {
            DateTime tt = DateTime.Now;
            label35.Text = Convert.ToString(tt.ToUniversalTime());
        }
        private void displayTable()
        {
            ArrayList tableList = new ArrayList(); //ArrayList to store list of tables
            table t = new table();
            tableList = t.readTableFileList(); //reads tables from file to list

            for (int i = 0; i < tableList.Count; i++) //checks each table
            {
                if (i == 0)
                {
                    label13.Text = ((tableList[i] as table).gameStatusProperty == 0) ? "✗" : "✓";
                    label23.Text = Convert.ToString((tableList[i] as table).gameStatusProperty);
                    label35.Text = ((tableList[i] as table).gameStatusProperty != 0)?Convert.ToString((tableList[i] as table).startTimeProperty.ToUniversalTime()):"-";
                }
                if (i == 1)
                {
                    label14.Text = ((tableList[i] as table).gameStatusProperty == 0) ? "✗" : "✓";
                    label24.Text = Convert.ToString((tableList[i] as table).gameStatusProperty);
                    label36.Text = ((tableList[i] as table).gameStatusProperty != 0)?Convert.ToString((tableList[i] as table).startTimeProperty.ToUniversalTime()):"-";
                }
                if (i == 2)
                {
                    label15.Text = ((tableList[i] as table).gameStatusProperty == 0) ? "✗" : "✓";
                    label25.Text = Convert.ToString((tableList[i] as table).gameStatusProperty);
                    label37.Text = ((tableList[i] as table).gameStatusProperty != 0)?Convert.ToString((tableList[i] as table).startTimeProperty.ToUniversalTime()):"-";
                }
                if (i == 3)
                {
                    label16.Text = ((tableList[i] as table).gameStatusProperty == 0) ? "✗" : "✓";
                    label26.Text = Convert.ToString((tableList[i] as table).gameStatusProperty);
                    label38.Text = ((tableList[i] as table).gameStatusProperty != 0)?Convert.ToString((tableList[i] as table).startTimeProperty.ToUniversalTime()):"-";
                }
                if (i == 4)
                        {
                            label17.Text = ((tableList[i] as table).gameStatusProperty == 0) ? "✗" : "✓";
                            label27.Text = Convert.ToString((tableList[i] as table).gameStatusProperty);
                            label39.Text = ((tableList[i] as table).gameStatusProperty != 0)?Convert.ToString((tableList[i] as table).startTimeProperty.ToUniversalTime()):"-";
```

```
                        }
                        if (i == 5)
                        {
                            label18.Text = ((tableList[i] as table).gameStatusProperty == 0) ? "✗" : "✓";
                            label28.Text = Convert.ToString((tableList[i] as table).gameStatusProperty);
                            label40.Text = ((tableList[i] as table).gameStatusProperty != 0)?Convert.ToString((tableList[i] as table).startTimeProperty.ToUniversalTime()):"-";
                        }
                        if (i == 6)
                        {
                            label19.Text = ((tableList[i] as table).gameStatusProperty == 0) ? "✗" : "✓";
                            label29.Text = Convert.ToString((tableList[i] as table).gameStatusProperty);
                            label41.Text = ((tableList[i] as table).gameStatusProperty != 0)?Convert.ToString((tableList[i] as table).startTimeProperty.ToUniversalTime()):"-";
                        }
                        if (i == 7)
                        {
                            label20.Text = ((tableList[i] as table).gameStatusProperty == 0) ? "✗" : "✓";
                            label30.Text = Convert.ToString((tableList[i] as table).gameStatusProperty);
                            label42.Text = ((tableList[i] as table).gameStatusProperty != 0)?Convert.ToString((tableList[i] as table).startTimeProperty.ToUniversalTime()):"-";
                        }
                        if (i == 8)
                        {
                            label21.Text = ((tableList[i] as table).gameStatusProperty == 0) ? "✗" : "✓";
                            label31.Text = Convert.ToString((tableList[i] as table).gameStatusProperty);
                            label43.Text = ((tableList[i] as table).gameStatusProperty != 0)?Convert.ToString((tableList[i] as table).startTimeProperty.ToUniversalTime()):"-";
                        }
                    }
                }

                private void button7_Click(object sender, EventArgs e)
                {
                    displayTable();
                    Application.Exit();
                }

                private void label1_Click(object sender, EventArgs e)
                {

                }
            }
        }
```

Form2.cs Code:

```
using System;
using System.Collections.Generic;
```

```
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace Assignment2
{
    public partial class Form2 : Form
    {
        public Form2()
        {
            InitializeComponent();
        }

        private void label3_Click(object sender, EventArgs e)
        {
                    }

        private void button1_Click(object sender, EventArgs e)
        {
            player p = new player();
            if (!p.createNewPlayer(double.Parse(textBox1.Text), textBox2.Text, text
Box3.Text, double.Parse(textBox4.Text)))
            {
                label6.Text = "ERROR! Player-ID already registered.";
            }
            else
            {
                label6.Text = "DONE! Player succesfully registered.";
            }
        }

        private void label6_Click(object sender, EventArgs e)
        {

        }
    }
}
```

Form3.cs Code:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace Assignment2
{
    public partial class Form3 : Form
    {
        public Form3()
        {
            InitializeComponent();
        }

        private void label2_Click(object sender, EventArgs e)
```

```
        {

        }

        private void button1_Click(object sender, EventArgs e)
        {
            player p = new player();
            if (radioButton3.Checked)
            {
                if(p.searchPlayer(textBox1.Text, 1))
                {
                    label10.Text = Convert.ToString(p.userIDProperty);
                    label11.Text = p.firstNameProperty+" "+p.lastNameProperty;
                    label12.Text = Convert.ToString(p.cnicProperty);
                    label13.Text = Convert.ToString(p.wonProperty);
                    label14.Text = Convert.ToString(p.drawProperty);
                    label15.Text = Convert.ToString(p.lostProperty );
                    label16.Text = "DONE! Player found.";
                }
                else
                {
                    label10.Text = "-";
                    label11.Text = "-";
                    label12.Text = "-";
                    label13.Text = "-";
                    label14.Text = "-";
                    label15.Text = "-";
                    label16.Text = "SORRY! Player not found.";
                }
            }
            if (radioButton2.Checked)
            {
                if (p.searchPlayer(textBox1.Text, 2))
                {
                    label10.Text = Convert.ToString(p.userIDProperty);
                    label11.Text = p.firstNameProperty + " " + p.lastNameProperty;

                    label12.Text = Convert.ToString(p.cnicProperty);
                    label13.Text = Convert.ToString(p.wonProperty);
                    label14.Text = Convert.ToString(p.drawProperty);
                    label15.Text = Convert.ToString(p.lostProperty);
                    label16.Text = "DONE! Player found.";
                }
                else
                {
                    label10.Text = "-";
                    label11.Text = "-";
                    label12.Text = "-";
                    label13.Text = "-";
                    label14.Text = "-";
                    label15.Text = "-";
                    label16.Text = "SORRY! Player not found.";
                }
            }
            if (radioButton1.Checked)
            {
                if (p.searchPlayer(textBox1.Text, 3))
                {
                    label10.Text = Convert.ToString(p.userIDProperty);
                    label11.Text = p.firstNameProperty + " " + p.lastNameProperty;

                    label12.Text = Convert.ToString(p.cnicProperty);
                    label13.Text = Convert.ToString(p.wonProperty);
                    label14.Text = Convert.ToString(p.drawProperty);
                    label15.Text = Convert.ToString(p.lostProperty);
                    label16.Text = "DONE! Player found.";
```

```
                }
                else
                {
                    label10.Text = "-";
                    label11.Text = "-";
                    label12.Text = "-";
                    label13.Text = "-";
                    label14.Text = "-";
                    label15.Text = "-";
                    label16.Text = "SORRY! Player not found.";
                }
            }
        }
    }
}
```

Form4.cs Code:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace Assignment2
{
    public partial class Form4 : Form
    {
        public Form4()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            table t = new table();
            if (t.assignNewTable(double.Parse(textBox1.Text)))
            {
                label3.Text = "DONE! Table assigned.";
            }
            else
            {
                label3.Text = "SORRY! Please wait, all tables are filled.";
            }
        }
    }
}
```

Form5.cs Code:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
```

```
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace Assignment2
{
    public partial class Form5 : Form
    {
        public Form5()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            table t = new table();
            if (t.assignNewTable(double.Parse(textBox1.Text), double.Parse(textBox2
.Text)))
            {
                label3.Text = "DONE! Table assigned.";
            }
            else
            {
                label3.Text = "SORRY! Please wait, all tables are filled.";
            }
        }
    }
}
```

Form6.cs Code:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace Assignment2
{
    public partial class Form6 : Form
    {
        public Form6()
        {
            InitializeComponent();
        }

        private void label2_Click(object sender, EventArgs e)
        {

        }

        private void button1_Click(object sender, EventArgs e)
        {
            table t = new table();
            t.submitTableResults(int.Parse(textBox1.Text), 1);
        }

        private void button1_Click_1(object sender, EventArgs e)
        {
```

```
            table t = new table();
            button2.Text = "Player-1: " + Convert.ToString(t.playerOneID(int.Parse(textBox1.Text)));
            button3.Text = "Player-2: " + Convert.ToString(t.playerTwoID(int.Parse(textBox1.Text)));
            int tab = int.Parse(textBox1.Text);
            if (tab == 1 || tab == 2 || tab == 3 || tab == 4 || tab == 5 || tab == 6 || tab == 7 || tab == 8 || tab == 9)
            {
                label3.Text = "DONE! Table found.";
            }
            else
            {
                label3.Text = "ERROR! Table not found.";
            }
        }

        private void button3_Click(object sender, EventArgs e)
        {
            table t = new table();
            t.submitTableResults(int.Parse(textBox1.Text), 2);
        }

        private void button4_Click(object sender, EventArgs e)
        {
            table t = new table();
            t.submitTableResults(int.Parse(textBox1.Text), 3);
        }
    }
}
```

Player.cs Code:

```
//by Muhammad Anas Baig-(01-134152-037)-BS(CS)-5A-VP
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.IO;
using System.Collections;

class player
    {
        double userID;
        string firstName;
        string lastName;
        double cnic;
        int won;
        int draw;
        int lost;

        public double userIDProperty
        {
            get{    return userID;  }
            set{    userID = value; }
        }
        public string firstNameProperty
        {
            get{    return firstName;   }
            set{    firstName = value;  }
        }
        public string lastNameProperty
```

```
        {
            get{    return lastName;    }
            set{    lastName = value;   }
        }
        public double cnicProperty
        {
            get{    return cnic;    }
            set{    cnic = value;   }
        }
        public int wonProperty
        {
            get{    return won;     }
            set{    won = value;    }
        }
        public int drawProperty
        {
            get {   return draw;    }
            set {   draw = value;   }
        }
        public int lostProperty
        {
            get{    return lost;    }
            set{    lost = value;   }
        }
        public player()
        {
            won = 0;
            draw = 0;
            lost = 0;
        }
        public string fullName(double id) //function that returns concatenation of firstname and lastname
        {
            ArrayList playerList = new ArrayList(); //player list
            playerList = readPlayerFile(); //reading file to list

            for (int i = 0; i < playerList.Count; i++) //checks each user in list
            {
                if ((playerList[i] as player).userID == id) //checks for the required user
                {
                    return ((playerList[i] as player).firstName + " " + (playerList[i] as player).lastName); //returns concatenation of firstname and lastname
                }
            }
            return (""); //if user not found
        }
        public bool createNewPlayer(double userID ,string firstName, string lastName, double cnic)
        {
            if (searchUniqueUserID(userID)) //not unique
            {
                return false;
            }
            else
            {
                this.userIDProperty = userID;
                this.firstNameProperty = firstName;
                this.lastNameProperty = lastName;
                this.cnicProperty = cnic;
                writePlayerFile(this); //appends the new player in player file
                return true;
            }
        }
        public bool searchUniqueUserID(double id) //before creating new user this methods checks either the userID is already taken or not
```

```
        {
            ArrayList playerList = new ArrayList(); //player list
            playerList = readPlayerFile(); //reading file to list

            for (int i = 0; i < playerList.Count; i++)  //checks each user in list
            {
                if ((playerList[i] as player).userID == id) //checks either userID is already taken or not
                {
                    return true;
                    }
                }
                return false;
            }
            public bool searchPlayer(string phrase, int type) //to search a specific user in the system
            {
                ArrayList playerList = new ArrayList(); //player list
                playerList = readPlayerFile(); //reading file to list

                if(type == 1) //search by userID
                {
                    double ID = double.Parse(phrase);

                    for (int i = 0; i < playerList.Count; i++) //checks each user in list
                    {
                        if ((playerList[i] as player).userID == ID) //checks for the requied userID
                        {
                            this.userIDProperty = (playerList[i] as player).userIDProperty;
                            this.firstNameProperty = (playerList[i] as player).firstNameProperty;
                            this.lastNameProperty = (playerList[i] as player).lastNameProperty;
                            this.cnicProperty = (playerList[i] as player).cnicProperty;
                            this.wonProperty = (playerList[i] as player).wonProperty;
                            this.drawProperty = (playerList[i] as player).drawProperty;
                            this.lostProperty = (playerList[i] as player).lostProperty;
                            return true;
                        }
                    }
                }
                if (type == 2) //search by name
                {
                    for (int i = 0; i < playerList.Count; i++) //checks each user in list
                    {
                        if (((playerList[i] as player).firstName + " " + (playerList[i] as player).lastName) == phrase) //checks for the requied name(firstName + lastName)
                        {
                            this.userIDProperty = (playerList[i] as player).userIDProperty;
                            this.firstNameProperty = (playerList[i] as player).firstNameProperty;
                            this.lastNameProperty = (playerList[i] as player).lastNameProperty;
                            this.cnicProperty = (playerList[i] as player).cnicProperty;
```

```
                                this.wonProperty = (playerList[i] as player).wonProperty;
                                this.drawProperty = (playerList[i] as player).drawProperty;
                                this.lostProperty = (playerList[i] as player).lostProperty;
                                return true;
                            }
                        }
                    }
                    if (type == 3) //search by cnic
                    {
                        double num = double.Parse(phrase);

                        for (int i = 0; i < playerList.Count; i++) //checks each user in list
                        {
                            if ((playerList[i] as player).cnic == num) //check for the required cnic
                            {
                                this.userIDProperty = (playerList[i] as player).userIDProperty;
                                this.firstNameProperty = (playerList[i] as player).firstNameProperty;
                                this.lastNameProperty = (playerList[i] as player).lastNameProperty;
                                this.cnicProperty = (playerList[i] as player).cnicProperty;
                                this.wonProperty = (playerList[i] as player).wonProperty;
                                this.drawProperty = (playerList[i] as player).drawProperty;
                                this.lostProperty = (playerList[i] as player).lostProperty;
                                return true;
                            }
                        }
                    }
                    return false;
                }
                public void displayAllPlayers() //to display all players statistics

                {
                    ArrayList playerList = new ArrayList(); //player list
                    playerList = readPlayerFile(); //reading file to list
                    Console.Write("------------------------------------------------------------------------------");
                    Console.WriteLine( "DISPLAY ALL PLAYERS STATISTICS:");
                    Console.Write("------------------------------------------------------------------------------");

                    for ( int i = 0; i < playerList.Count; i++ ) //checks each user in list
                    {
                        Console.WriteLine();
                        Console.WriteLine("----------------------");
                        Console.WriteLine("Player - " + (i+1) + " Statistics:");
                        Console.WriteLine("----------------------" );
                        Console.WriteLine("User ID:     " + (playerList[i] as player).userID);
                        Console.WriteLine("Name:        " + (playerList[i] as player).firstName + " " + (playerList[i] as player).lastName);
                        Console.WriteLine("CNIC:        " + (playerList[i] as player).cnic);
                        int gamesPlayed = ((playerList[i] as player).won + (playerList[i] as player).draw + (playerList[i] as player).lost); //number of games played
```

```
                        Console.WriteLine("SCORE:");
                        Console.WriteLine("------");
                        Console.WriteLine("Won:         " + (playerList[i] as player ).won);
                        Console.WriteLine("Draw:        " + (playerList[i] as player ).draw);
                        Console.WriteLine("Lost:        " + (playerList[i] as player ).lost);
                        Console.WriteLine("----------------------");
                        Console.WriteLine("Total Games: " + gamesPlayed);
                        Console.WriteLine("----------------------");
                    }
                }
                public ArrayList readPlayerFile() //reads player file and returns ArrayList which containts all players data
                {
                    ArrayList playerList = new ArrayList(); //to display all players statistics
                    StreamReader readPlayerFile = new StreamReader("Players.txt"); //reading file to list
                    player p;

                    while (!readPlayerFile.EndOfStream) //reading file till end
                    {
                        p = new player();
                        p.userID = double.Parse(readPlayerFile.ReadLine());
                        p.firstName = readPlayerFile.ReadLine();
                        p.lastName = readPlayerFile.ReadLine();
                        p.cnic = double.Parse(readPlayerFile.ReadLine());
                        p.won = int.Parse(readPlayerFile.ReadLine());
                        p.draw = int.Parse(readPlayerFile.ReadLine());
                        p.lost = int.Parse(readPlayerFile.ReadLine());
                        playerList.Add( p );
                    }
                    readPlayerFile.Close();
                    return playerList; //returning ArrayList which contains all players data
                }
                public void writePlayerFile(player p) //to add new player to the file
                {
                    StreamWriter writePlayerFile = new StreamWriter("Players.txt", true); //appending the player file

                    writePlayerFile.WriteLine( p.userID );
                    writePlayerFile.WriteLine( p.firstName );
                    writePlayerFile.WriteLine( p.lastName );
                    writePlayerFile.WriteLine( p.cnic );
                    writePlayerFile.WriteLine( p.won );
                    writePlayerFile.WriteLine( p.draw );
                    writePlayerFile.WriteLine( p.lost );

                    writePlayerFile.Close();
                }
                public void writePlayerFileList(ArrayList playerList) //to write modified/updated data to file -> modify/update -> game Win/Loss
                {
                    StreamWriter writePlayerFile = new StreamWriter("Players.txt"); //not opened in appended mode because all modified/updated data is to write to file -> modify/update -> game Win/Loss

                    for (int i = 0; i < playerList.Count; i++) //checks each user in list
                    {
                        writePlayerFile.WriteLine((playerList[i] as player ).userIDProperty);
```

```
                        writePlayerFile.WriteLine((playerList[i] as player).firstNameProperty);
                        writePlayerFile.WriteLine((playerList[i] as player).lastNameProperty);
                        writePlayerFile.WriteLine((playerList[i] as player).cnicProperty);
                        writePlayerFile.WriteLine((playerList[i] as player).wonProperty);
                        writePlayerFile.WriteLine((playerList[i] as player).drawProperty);
                        writePlayerFile.WriteLine((playerList[i] as player).lostProperty);
                    }
                    writePlayerFile.Close();
                }
                public void playerWon(double id) //takes userID and updates user's won games
                {
                    ArrayList playerList = new ArrayList(); //player list
                    playerList = readPlayerFile(); //reading file to list

                    for (int i = 0; i < playerList.Count; i++) //checks each user in list
                    {
                        if ((playerList[i] as player).userIDProperty == id)
                        {
                            (playerList[i] as player).wonProperty = 1 + (playerList[i] as player).wonProperty; //increments in user's won games
                        }
                    }
                    writePlayerFileList(playerList);
                }
                public void playerLost(double id) //takes userID and updates user's won games
                {
                    ArrayList playerList = new ArrayList(); //player list
                    playerList = readPlayerFile(); //reading file to list

                    for (int i = 0; i < playerList.Count; i++) //checks each user in list
                    {
                        if ((playerList[i] as player).userIDProperty == id) //checks if required user is found
                        {
                            (playerList[i] as player).lostProperty = 1 + (playerList[i] as player).lostProperty; //increments in user's won games
                        }
                    }
                    writePlayerFileList(playerList);
                }
                public void playerDraw(double id1, double id2) //takes userIDs of player1 and player2 and updates both user's draw games
                {
                    ArrayList playerList = new ArrayList(); //player list
                    playerList = readPlayerFile(); //reading file to list

                    for (int i = 0; i < playerList.Count; i++) //checks each user in list
                    {
                        if ((playerList[i] as player).userIDProperty == id1)
                        {
                            (playerList[i] as player).drawProperty = 1 + (playerList[i] as player).drawProperty; //increments player's draw games
                        }
                        if ((playerList[i] as player).userIDProperty == id2)
                        {
```

```
                            (playerList[i] as player).drawProperty = 1 + (playerList[i] as player).drawProperty; //increments player's draw games
                        }
                    }
                    writePlayerFileList( playerList );
                }
            }
        //by Muhammad Anas Baig-(01-134152-037)-BS(CS)-5A-VP
```

Tables.cs Code:

```
//by Muhammad Anas Baig-(01-134152-037)-BS(CS)-5A-VP
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.IO;
using System.Collections;

class table
{
    int tableID; //stores tableID
    int gameStatus; //stores game status i.e. 0->Empty Table, 1->One Player Assigned, 2->Two Players Assigned
    DateTime startTime;
    DateTime endTime;
    public player playerOne = new player(); //player1 on table
    public player playerTwo = new player(); //player2 on table

    public int tableIDProperty
    {
        get { return tableID; }
        set { tableID = value; }
    }
    public int gameStatusProperty
    {
        get { return gameStatus; }
        set { gameStatus = value; }
    }
    public double playerOneProperty
    {
        get { return playerOne.userIDProperty; }
        set { playerOne.userIDProperty = value; }
    }
    public double playerTwoProperty
    {
        get { return playerTwo.userIDProperty; }
        set { playerTwo.userIDProperty = value; }
    }
    public DateTime startTimeProperty
    {
        get { return startTime; }
        set { startTime = value; }
    }
    public DateTime endTimeProperty
    {
        get { return endTime; }
        set { endTime = value; }
    }
    public bool searchUniqueTableID(int id) //while creating new table checks either the tableID in already assigned or not
    {
```

```
        ArrayList tableList = new ArrayList(); //ArrayList to store list of tables

        tableList = readTableFileList(); //reads tables from file to list

        for (int i = 0; i < tableList.Count; i++) //checks each table
        {
            if ((tableList[i] as table).tableID == id) //checks for the required tableID
            {
                return true;
            }
        }
        return false;
    }
    public void createNewTable() //to add new table to system
    {
        Console.Write("------------------------------------------------------------------------------------");
        Console.WriteLine("ADD NEW TABLE TO SYSTEM:");
        Console.Write("------------------------------------------------------------------------------------");

        do
        {
            Console.WriteLine("Enter New Table-ID:");
            this.tableID = int.Parse(Console.ReadLine());
            if (searchUniqueTableID(this.tableID)) //while creating new table checks either the tableID in already assigned or not
            {
                Console.WriteLine("ERROR! Table-ID already assigned, kindly choose another.");
            }
        }
        while (searchUniqueTableID(this.tableID));

        gameStatusProperty = 0; //game status i.e. 0->Empty Table, 1->One Player Assigned, 2->Two Players Assigned
        playerOneProperty = 0;
        playerTwoProperty = 0;
        startTimeProperty = DateTime.Now;
        endTimeProperty = DateTime.Now;
        Console.WriteLine("Table Successfully Created.");

        writeAddTableFile(this); //appends new to table to table file
    }
    public ArrayList readTableFileList() //reads table file to list and then returns list
    {
        ArrayList tableList = new ArrayList(); //ArrayList to store list of tables

        StreamReader readTableFile = new StreamReader("Tables.txt"); //read file
        table t;

        while (!readTableFile.EndOfStream) //reads table file till end
        {
            t = new table();
            t.tableIDProperty = int.Parse(readTableFile.ReadLine());
            t.gameStatusProperty = int.Parse(readTableFile.ReadLine());
            t.startTime = DateTime.Parse(readTableFile.ReadLine());
                t.endTime = DateTime.Parse(readTableFile.ReadLine());
                t.playerOneProperty = double.Parse(readTableFile.ReadLine());
                t.playerTwoProperty = double.Parse(readTableFile.ReadLine());
                tableList.Add(t);
            }
            readTableFile.Close();
            return tableList;
```

```
        }
        public void writeAddTableFile(table t) //to add new table to the sysem by appending
        {
            StreamWriter writeTableFile = new StreamWriter("Tables.txt", true); //appending table file

            writeTableFile.WriteLine(t.tableIDProperty);
            writeTableFile.WriteLine(t.gameStatusProperty);
            writeTableFile.WriteLine(t.startTimeProperty);
            writeTableFile.WriteLine(t.endTimeProperty);
            writeTableFile.WriteLine(t.playerOneProperty);
            writeTableFile.WriteLine(t.playerTwoProperty);

            writeTableFile.Close();
        }
        public void writeTableFileList(ArrayList tableList) //to write modified/updated data to file -> modify/update -> table status
        {
            StreamWriter writeTableFile = new StreamWriter("Tables.txt"); //not opened in appended mode because all modified/updated data is to write to file -> modify/update -> table status

            for (int i = 0; i < tableList.Count; i++)
            {
                writeTableFile.WriteLine((tableList[i] as table).tableIDProperty);
                writeTableFile.WriteLine((tableList[i] as table).gameStatusProperty);
                writeTableFile.WriteLine((tableList[i] as table).startTimeProperty);
                writeTableFile.WriteLine((tableList[i] as table).endTimeProperty);
                writeTableFile.WriteLine((tableList[i] as table).playerOneProperty);
                writeTableFile.WriteLine((tableList[i] as table).playerTwoProperty);
            }
            writeTableFile.Close();
        }
        public bool assignNewTable(double playerOneUserID) //to assign new table if one player comes
        {
            ArrayList tableList = new ArrayList(); //ArrayList to store list of tables
            tableList = readTableFileList(); //reads tables from file to list

            for (int i = 0; i < tableList.Count; i++) //checks each table
            {
                if ((tableList[i] as table).gameStatus == 0) //if table is empty

                {
                    (tableList[i] as table).gameStatusProperty = 1; //game status i.e. 0->Empty Table, 1->One Player Assigned, 2->Two Players Assigned
                    (tableList[i] as table).startTimeProperty = DateTime.Now;
                    (tableList[i] as table).endTimeProperty = DateTime.Now;
                    (tableList[i] as table).playerOneProperty = playerOneUserID;

                    (tableList[i] as table).playerTwoProperty = 0;
                    writeTableFileList(tableList); //write again to file
                    return true;
                }
                else if ((tableList[i] as table).gameStatus == 1) //if table has 1 playyer then assign the new player to this table
                {
```

```
                        (tableList[i] as table).gameStatusProperty = 2; //game status i.e. 0->Empty Table, 1->One Player Assigned, 2->Two Players Assigned
                        (tableList[i] as table).startTimeProperty = DateTime.Now;
                        (tableList[i] as table).endTimeProperty = DateTime.Now;
                        (tableList[i] as table).playerTwoProperty = playerOneUserID;

                        writeTableFileList(tableList); //write again to file
                        return true;
                    }
                }
                return false; //all tables filled
            }
            public bool assignNewTable(double playerOneUserID, double playerTwoUserID) //to assign new table if two players come
            {
                ArrayList tableList = new ArrayList(); //ArrayList to store list of tables
                tableList = readTableFileList(); //reads tables from file to list

                for (int i = 0; i < tableList.Count; i++) //checks each table
                {
                    if ((tableList[i] as table).gameStatus == 0) //if table is empty then assign to them
                    {
                        (tableList[i] as table).gameStatusProperty = 2; //game status i.e. 0->Empty Table, 1->One Player Assigned, 2->Two Players Assigned
                        (tableList[i] as table).startTimeProperty = DateTime.Now;
                        (tableList[i] as table).endTimeProperty = DateTime.Now;
                        (tableList[i] as table).playerOneProperty = playerOneUserID;

                        (tableList[i] as table).playerTwoProperty = playerTwoUserID;

                        writeTableFileList(tableList); //write again to file
                        Console.WriteLine("Table Successfully Assigned.");
                        return true;
                    }
                }
                for (int i = 0; i < tableList.Count; i++) //checks each table
                {
                    if ((tableList[i] as table).gameStatus == 1) //as no full table is empty so now it will check table where one player is assigned so that they can start game immediately
                    {
                        (tableList[i] as table).gameStatusProperty = 2; //game status i.e. 0->Empty Table, 1->One Player Assigned, 2->Two Players Assigned
                        (tableList[i] as table).startTimeProperty = DateTime.Now;
                        (tableList[i] as table).endTimeProperty = DateTime.Now;
                        (tableList[i] as table).playerOneProperty = playerOneUserID;

                        (tableList[i] as table).playerTwoProperty = playerTwoUserID;

                        writeTableFileList(tableList);
                        Console.WriteLine("Table Successfully Assigned.");
                        return true;
                    }
                }
                return false;
            }
            public void displayTableList() //to display all tables status
            {
                ArrayList tableList = new ArrayList(); //ArrayList to store list of tables
                tableList = readTableFileList(); //reads tables from file to list

                Console.Write("------------------------------------------------------------------------------");
```

```
                Console.WriteLine("DISPLAY ALL TABLES STATUS:");
                Console.Write("-------------------------------------------------------------------------------");

                for (int i = 0; i < tableList.Count; i++) //checks each table
                {
                    Console.WriteLine();
                    Console.WriteLine("----------------------");
                    Console.WriteLine("Table-ID: " + (tableList[i] as table).tableID);
                    Console.WriteLine("----------------------");
                    if ((tableList[i] as table).gameStatus == 0) //game status i.e. 0->Empty Table, 1->One Player Assigned, 2->Two Players Assigned
                    {
                        Console.WriteLine("0 Players Assigned.");
                    }
                    else if ((tableList[i] as table).gameStatus == 1) //game status i.e. 0->Empty Table, 1->One Player Assigned, 2->Two Players Assigned
                    {
                        player p = new player();
                        Console.WriteLine("1 Player Assigned.");

                        string playerOneName = p.fullName((tableList[i] as table).playerOneProperty);
                        Console.WriteLine("1. Player-1 (ID-" + (tableList[i] as table).playerOneProperty + ") " + playerOneName);
                        Console.WriteLine("Start Time:" + (tableList[i] as table).startTimeProperty);
                    }
                    if ((tableList[i] as table).gameStatus == 2) //game status i.e. 0->Empty Table, 1->One Player Assigned, 2->Two Players Assigned
                    {
                        player p = new player();
                        Console.WriteLine("2 Players Assigned.");

                        string playerOneName = p.fullName((tableList[i] as table).playerOneProperty);
                        string playerTwoName = p.fullName((tableList[i] as table).playerTwoProperty);
                        Console.WriteLine("1. Player-1 (ID-" + (tableList[i] as table).playerOneProperty + ") " + playerOneName);
                        Console.WriteLine("2. Player-2 (ID-" + (tableList[i] as table).playerTwoProperty + ") " + playerTwoName);
                        Console.WriteLine("Start Time:" + (tableList[i] as table).startTimeProperty);
                    }
                }
            }
            public void submitTableResults(int id, int result) //to submit game results and clear table status
            {
                ArrayList tableList = new ArrayList(); //ArrayList to store list of tables
                tableList = readTableFileList(); //reads tables from file to list

                for (int i = 0; i < tableList.Count; i++) //checks each table
                {
                    if ((tableList[i] as table).tableID == id) //checks for the required table
                    {
                        player p = new player();
                        if (result == 1)
                        {
                            p.playerWon((tableList[i] as table).playerOneProperty); //updates won status of player1
```

```
                            p.playerLost((tableList[i] as table).playerTwoProperty); //updates lost status of player2
                        }
                        else if (result == 2)
                        {
                            p.playerWon((tableList[i] as table).playerTwoProperty); //updates won status of player2
                            p.playerLost((tableList[i] as table).playerOneProperty); //updates lost status of player1
                        }
                        else if (result == 3)
                        {
                            p.playerDraw((tableList[i] as table).playerOneProperty, (tableList[i] as table).playerTwoProperty); //updates draw status of both users
                        }
                        (tableList[i] as table).endTimeProperty = DateTime.Now;

                        StreamWriter writeGameLogFile = new StreamWriter("GameLog.txt", true);
                        writeGameLogFile.WriteLine((tableList[i] as table).tableIDProperty);
                        writeGameLogFile.WriteLine((tableList[i] as table).playerOneProperty);
                        writeGameLogFile.WriteLine((tableList[i] as table).playerTwoProperty);
                        writeGameLogFile.WriteLine((tableList[i] as table).startTimeProperty);
                        writeGameLogFile.WriteLine((tableList[i] as table).endTimeProperty);
                        writeGameLogFile.Close();

                        (tableList[i] as table).gameStatus = 0; //clears table status to empty
                        (tableList[i] as table).playerOneProperty = 0; //clears table player1 to empty
                        (tableList[i] as table).playerTwoProperty = 0; //clears table player2 to empty
                        //(tableList[i] as table).startTimeProperty = null; //clears table player2 to empty
                        //(tableList[i] as table).playerTwoProperty = 0; //clears table player2 to empty
                        writeTableFileList(tableList);
                        return;
                    }
                }
            }
            public double playerOneID(int id) //to submit game results and clear table status
            {
                ArrayList tableList = new ArrayList(); //ArrayList to store list of tables
                tableList = readTableFileList(); //reads tables from file to list

                for (int i = 0; i < tableList.Count; i++) //checks each table
                {
                    if ((tableList[i] as table).tableID == id) //checks for the required table
                    {
                        player p = new player();
                        return ((tableList[i] as table).playerOneProperty);
                    }
                }
                return 0;
            }
            public double playerTwoID(int id) //to submit game results and clear table status
```

```
        {
            ArrayList tableList = new ArrayList(); //ArrayList to store list of tables
            tableList = readTableFileList(); //reads tables from file to list

            for (int i = 0; i < tableList.Count; i++) //checks each table
            {
                if ((tableList[i] as table).tableID == id) //checks for the required table
                {
                    player p = new player();
                    return ((tableList[i] as table).playerTwoProperty);
                }
            }
            return 0;
        }
        public void displayGameLogFile() //to display game history
        {
            Console.Write("------------------------------------------------------------------------------");
            Console.WriteLine("DISPLAY GAME LOG HISTORY:");
            Console.Write("------------------------------------------------------------------------------");

            double id;
            double player1ID;
            double player2ID;
            DateTime start;
            DateTime end;

            StreamReader readGameLogFile = new StreamReader("GameLog.txt");
            while (!readGameLogFile.EndOfStream) //reads game log file till end

            {
                id = double.Parse(readGameLogFile.ReadLine());
                player1ID = double.Parse(readGameLogFile.ReadLine());
                player2ID = double.Parse(readGameLogFile.ReadLine());
                start = DateTime.Parse(readGameLogFile.ReadLine());
                end = DateTime.Parse(readGameLogFile.ReadLine());

                Console.WriteLine("T-ID:" + id + " | P1-ID:" + player1ID + " | P2-ID:" + player2ID + " | START:" + start + " | END:" + end);
            }
        }
    }
    //by Muhammad Anas Baig-(01-134152-037)-BS(CS)-5A-VP
```

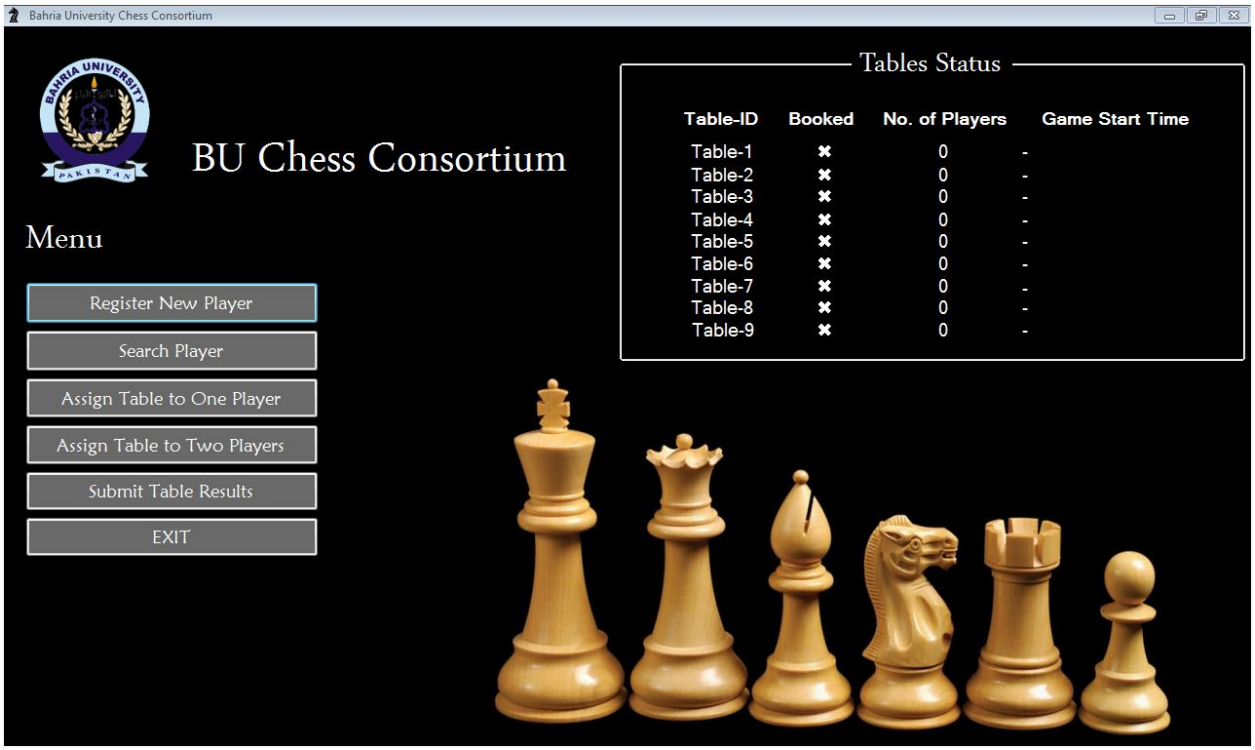
Program.cs Code:

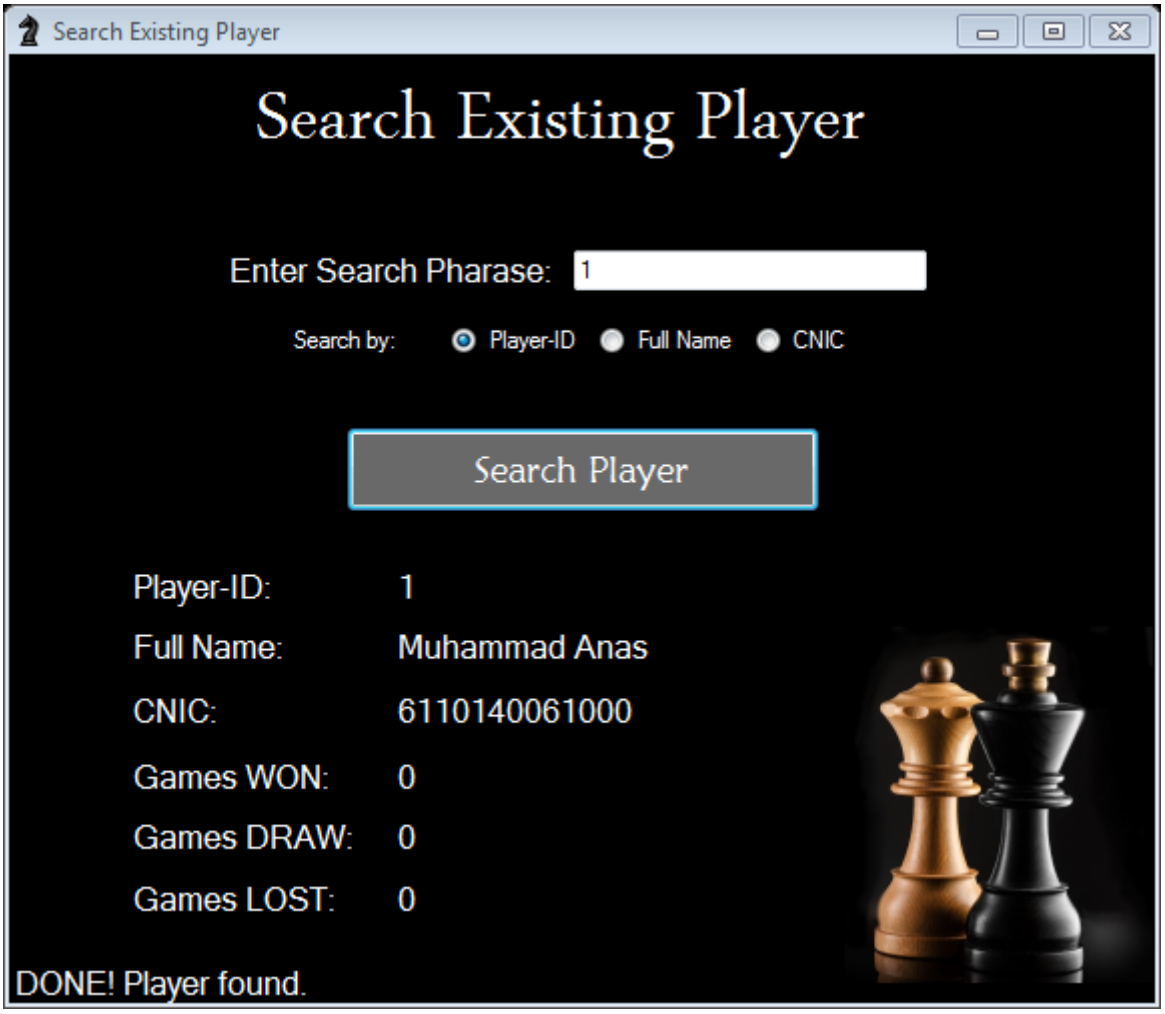
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Windows.Forms;

namespace Assignment2
{
    static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
```

```
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Form1());
        }
    }
}
```

Windows Form Application Output:



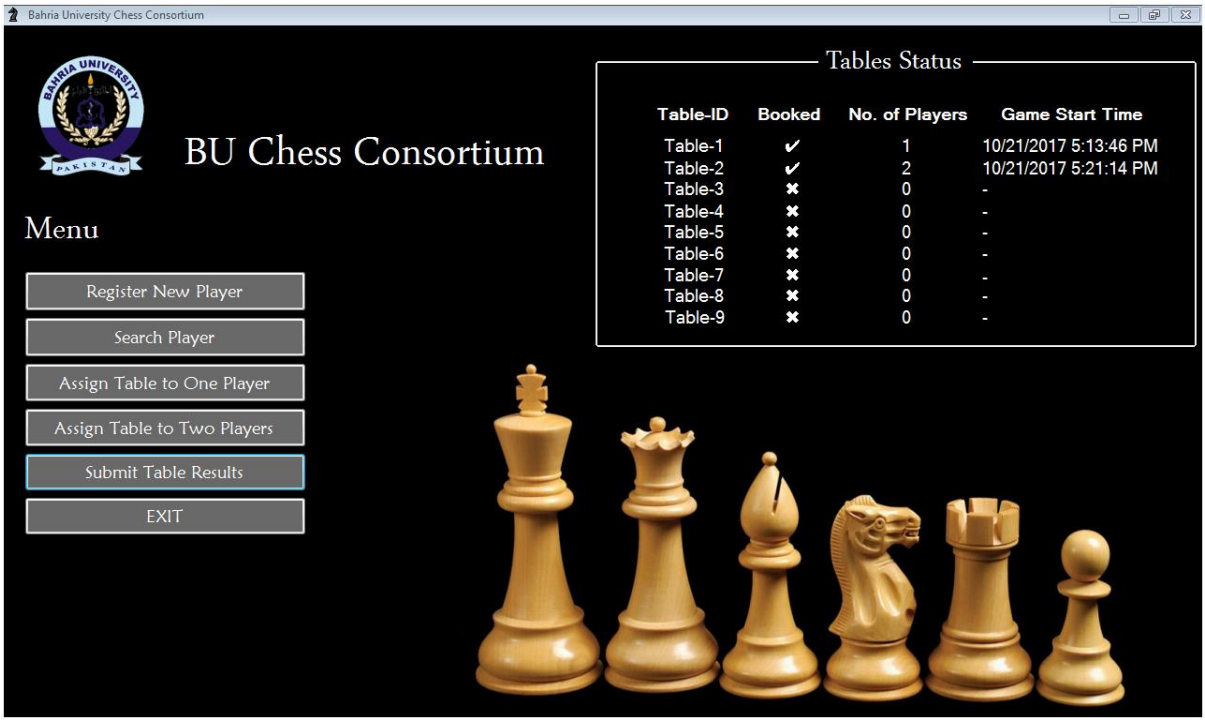

Search Existing Player
Search Existing Player
Enter Search Pharase:
1
Search by:
Player-ID
Full Name
CNIC
Search Player
Player-ID:
1
Full Name:
Muhammad Anas
CNIC:
6110140061000
Games WON:
0
Games DRAW:
0
Games LOST:
0
DONE! Player found.


Assign Table to One Player
Assign Table to One Player
Enter Player-ID:
1
Assign Table
DONE! Table assigned.

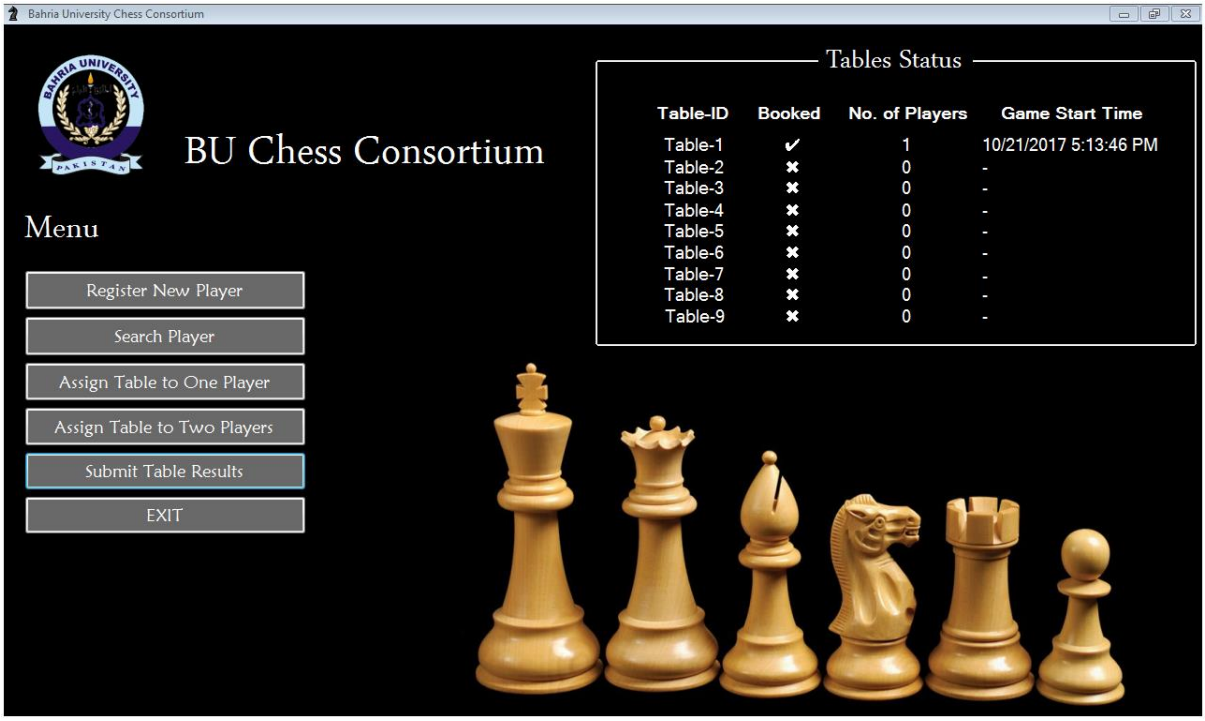

New Player Registration
New Player Registration
Enter Player-ID
2
Enter First Name:
Ali
Enter Last Name:
Mirza
Enter CNIC"
6110193939393
Register New Player
DONE! Player succesfully registered.


New Player Registration
New Player Registration
Enter Player-ID
3
Enter First Name:
Imran
Enter Last Name:
Siddiqui
Enter CNIC"
6110193944444
Register New Player
DONE! Player succesfully registered.


Assign Table to Two Players
Assign Table to Two Players
Enter Player-1 User-ID:
2
Enter Player-2 User-ID:
3
Assign Table
DONE! Table assigned.

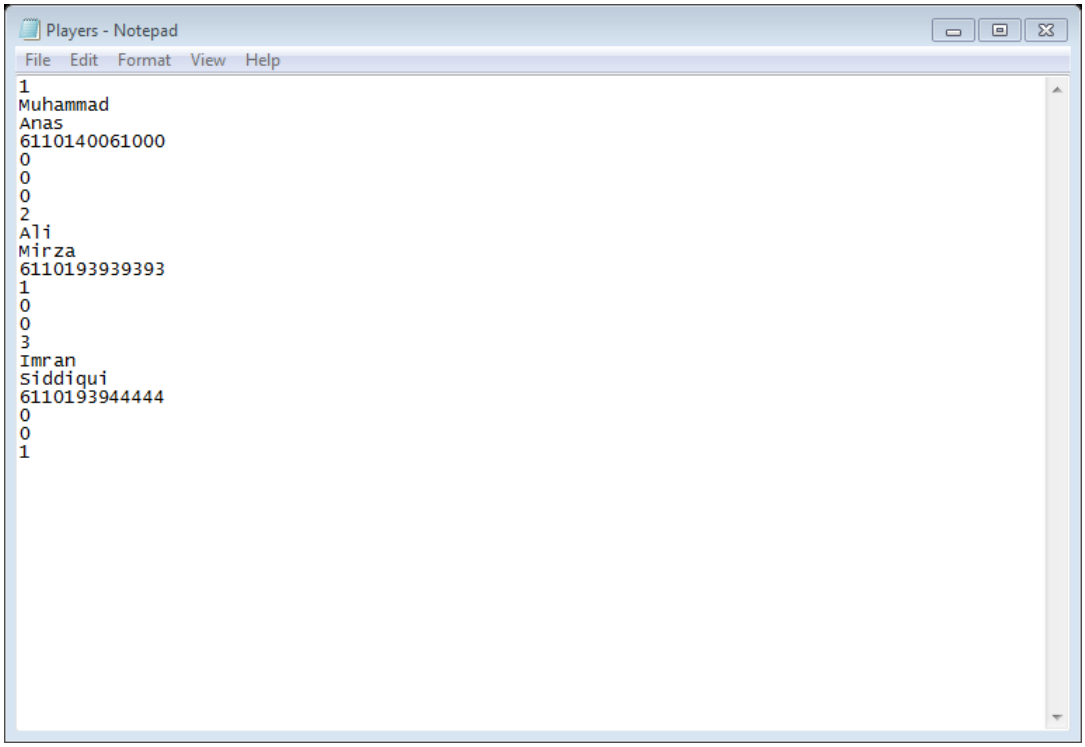

Bahria University Chess Consortium
BU Chess Consortium
Menu
Register New Player
Search Player
Assign Table to One Player
Assign Table to Two Players
Submit Table Results
EXIT
Tables Status
Table-ID Booked No. of Players Game Start Time
Table-1 ✔ 1 10/21/2017 5:13:46 PM
Table-2 ✔ 2 10/21/2017 5:21:14 PM
Table-3 ✖ 0 -
Table-4 ✖ 0 -
Table-5 ✖ 0 -
Table-6 ✖ 0 -
Table-7 ✖ 0 -
Table-8 ✖ 0 -
Table-9 ✖ 0 -


Submit Table Results
Submit Table Results
Enter Table-ID:
2
Submit
Player-1: 2
Player-2: 3
Game Draw
DONE! Table found.


Bahria University Chess Consortium
BU Chess Consortium
Menu
Register New Player
Search Player
Assign Table to One Player
Assign Table to Two Players
Submit Table Results
EXIT
Tables Status
Table-ID Booked No. of Players Game Start Time
Table-1 ✔ 1 10/21/2017 5:13:46 PM
Table-2 ✖ 0 -
Table-3 ✖ 0 -
Table-4 ✖ 0 -
Table-5 ✖ 0 -
Table-6 ✖ 0 -
Table-7 ✖ 0 -
Table-8 ✖ 0 -
Table-9 ✖ 0 -

Text File Output:

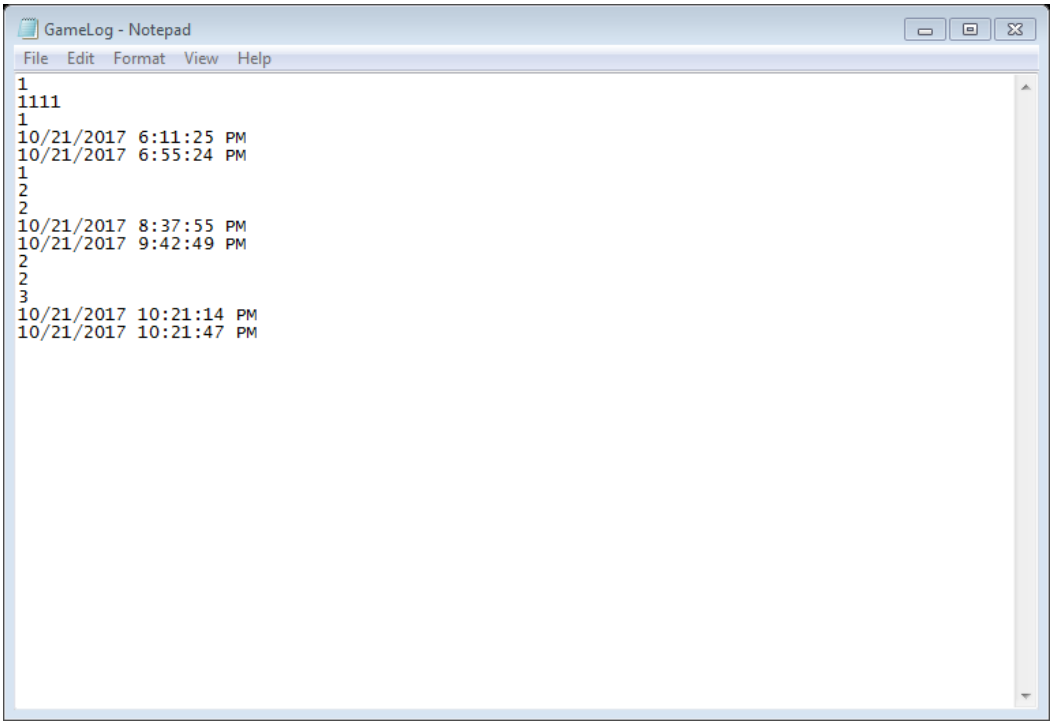
Players.txt File:


Players - Notepad

File Edit Format View Help

```
1
Muhammad
Anas
6110140061000
0
0
0
2
Ali
Mirza
6110193939393
1
0
0
3
Imran
Siddiqui
6110193944444
0
0
1
```

GameLog.txt File:


GameLog - Notepad

File Edit Format View Help

```
1
1111
1
10/21/2017 6:11:25 PM
10/21/2017 6:55:24 PM
1
2
2
10/21/2017 8:37:55 PM
10/21/2017 9:42:49 PM
2
2
3
10/21/2017 10:21:14 PM
10/21/2017 10:21:47 PM
```

Tables.txt File:

Tables - Notepad

File Edit Format View Help

```
1
1
10/21/2017 10:13:46 PM
10/21/2017 10:13:46 PM
1
0
2
0
10/21/2017 10:21:14 PM
10/21/2017 10:21:47 PM
0
0
3
0
10/21/2017 10:19:26 PM
10/21/2017 10:19:26 PM
0
0
4
0
10/21/2017 10:19:32 PM
10/21/2017 10:19:32 PM
```