```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.IO;
using System.Threading;

namespace Millionare
{
    public struct names
    {
        public string fname;
        public string lname;
        public string inter;
    }
    public struct questionare
    {
        public string des1;
        public string questions;
        public string emptyline;
        public string a;
        public string b;
        public string c;
        public string d;
        public string ans;
        public string des2;
    }
    public struct money
    {
        public string reward;
```

```
}
class Millioanre
{
    public static bool menuLoop = true,gameloop=true;
    static void Main()
    {
        Console.SetWindowSize(160, 40);                 //Sets window size
        Console.OutputEncoding = System.Text.Encoding.UTF8; //unicode used for desgin
        questionare[] answers = new questionare[17];       //array used for questionare struct
        money[] rewards = new money[16];                 //array used for moeny struct
        string[] Chosen = new string[1];                 //array used to for the top1 finalist
        string[] Finalists = new string[10];             //array used for top10 finalists
        names[] People = new names[30];                  //array used to names struct
        welcome();                                  //welcome
        read(People);                                //reads millionare txt
        sort(People);                               //sorts ppl from last name to first name
        readingquestions(answers);                       //reads questions
        readingmoney(rewards);                          //reads money
        options();                                  //displays options
        menu(People, Finalists, Chosen, answers,rewards);   //displays menu
        Top10(People);
        Console.ReadLine();
    }
    public static void welcome()
    {
        Console.Clear();
        Console.WriteLine("                  ___ _              _____     _     _        _     _     ");
        Console.WriteLine("                 / __\\ | |             |  __ \\   | |    | |      | |     (_)
");
        Console.WriteLine("                 | |  | | |_ __ _ __ _ ___   | |__) |__ | |_  _| |_ ___ ___|
|__ _ __ _ ___ ");
```

```
        Console.WriteLine("                    | |  | | __/ _` |/ _` |/ _ \\  |  ___/ _ \\| | | | | __/ _ \\/ __|
'_ \\| '_ \\| |/ __|");

        Console.WriteLine("                    | |__| | || (_| | (_| | (_) | | |  | (_) | | |_| | ||  __/ (__| |
| | | | | | (__ ");

        Console.WriteLine("                     \\_____/ \\__\\__,_|\\__, |\\___/  |_|  \\___/|_|\\__,
|\\__\\___|\\___|_| |_|_| |_|_|\\___|");

        Console.WriteLine("                                  __/ |                __/ |                      ");

        Console.WriteLine("                                 |___/                |___/                       ");

        Console.WriteLine("");

        Console.WriteLine("                _____ _   _ _____ __  ___                          _
_   ");

        Console.WriteLine("               |_  _| \\ | | ____/_ |/ _ \\               /\\         (_)
| |  ");

        Console.WriteLine("                 | | |  \\| | |__  | | | | |             /  \\  ___ ___ _ __ _ _ __  _
__ ___   ___ _ __ | |_ ");

        Console.WriteLine("                 | | | . ` |___ \\ | | | | |            / /\\ \\ / __/ __| |/ _` | '_ \\|
'_ ` _ \\ / _ \\ '_ \\| __|");

        Console.WriteLine("                _| |_| |\\  |___) || | |_| |           / ____ \\__ \\__ \\ | (_| |
| | | | | | | |  __/| | | |_ ");

        Console.WriteLine("               |_____|_| \\_|____/ |_|\\___/           /_/
\\_\\___/___/_|\\__, |_| |_|_| |_| |_|\\___|_| |_|\\__|");

        Console.WriteLine("                                                      __/ |                      ");

        Console.WriteLine("                                                     |___/                       ");

        Console.WriteLine("");

        Console.WriteLine(" __        ___                      _       _       _             __ __ _ _ _ _
_         ___  ");

        Console.WriteLine(" \\ \\      / / |                    | |     | |     | |           |  \\/  (_) | (_)
(_)       |__ \\ ");

        Console.WriteLine("  \\ \\ /\\ / /| |__   ___  __    _____ _ _ __ | |_ ___  | |_ ___  | |__   ___
__ _  | \\  / |_| | |_ ___ _ __  __ _ _ __ ___    ) |");

        Console.WriteLine("   \\ \\/  \\/ / | '_ \\ / _ \\ \\ \\ /\\ / / _` | '_ \\| __/ __| | __/ _ \\ | '_ \\ /
_ \\ / _` | | |\\/| | | | | |/ _ \\| '_ \\ / _` | | '__/ _ \\  / / ");

        Console.WriteLine("    \\  /\\  /  | | | | (_) |  \\ V  V / (_| | | | | |_\\__ \\ | || (_) | | |_) |  __/
| (_| | | |  | | | | | | (_) | | | | (_| | | | |  __/  |_|  ");

        Console.WriteLine("     \\/  \\/   |_| |_|\\___/   \\_/\\_/ \\__,_|_| |_|\\__|___/ \\__\\___/
|_.__/ \\___|  \\__,_| |_|  |_|_|_|_|_|\\___/|_| |_|\\__,_|_|_|_|  \\___|  (_)  ");
```

```
        Console.WriteLine("");
        Console.WriteLine("                    ___ ___       _     _         _ _    _          _ ");
        Console.WriteLine("                    |  \\/  |       | |   | |       | \\ | |   | |        | |");
        Console.WriteLine("                    | .  . | __ _ __| | ___  | |__ _  _  |  \\| | __ _| |__
___  ___| |");
        Console.WriteLine("                    | |\\/| |/ _` |/ _` |/ _ \\ | '_ \\| | | | | . ` |/ _` | '_ \\ / _
\\/ _ \\ |");
        Console.WriteLine("                    | |  | | (_| | (_| |  __/ | |_) | |_| | | |\\  | (_| | |_) |
__/  __/ |");
        Console.WriteLine("                    \\_|  |_/\\__,_|\\__,_|\\___| |_.__/ \\__, | \\_|
\\_/\\__,_|_.__/ \\___|\\___|_|");
        Console.WriteLine("                                          __/ |                    ");
        Console.WriteLine("                                         |___/                     ");
        Console.WriteLine("");
        Thread.Sleep(1);
        Console.Clear();
    }
    public static void menu(names[] People, string[] Finalists, string[] Chosen, questionare[]
answers, money[] rewards)
    {
        while (menuLoop)
        {
            Console.Clear();
            Console.WriteLine("");
            Console.WriteLine("");
            Console.WriteLine("");
            Console.WriteLine("     __       ___                    _      _       _            __ __ _ _ _ _
_         ___  ");
            Console.WriteLine("     \\\ \\\      / / |                  | |     | |     | |           | \\/ (_) |
(_)           (_)       |__ \\ ");
            Console.WriteLine("      \\\ \\\  /\\\  / /| |__  ___  __    ____ _ _ __ | |_ ___  | |_ ___  | |__
___   __ _  | \\  / |_| | |_ ___ _ __  __ _ _ _ _ __ ___    ) |");
            Console.WriteLine("       \\\ \\\/  \\\/ / | '_ \\\ / _ \\\  \\\ \\\ /\\\ / / _` | '_ \\\| __/ __| | __/ _ \\\  |
'_ \\\ / _ \\\  / _` | | |\\\/| | | | | |/ _ \\\| '_ \\\ / _` | | '__/ _ \\\  / / ");
```

```
        Console.WriteLine("       \\  /\\  /  | | | | (_) |  \\ V  V / (_| | | | | |_\\__ \\ | || (_) | | |_) |
__/ | (_| | | |  | | | | | | (_) | | | | (_| | | | |  __/  |_|  ");

        Console.WriteLine("       \\/  \\/   |_| |_|\\___/   \\_/\\_/ \\__,_|_| |_|\\__|___/
\\__\\___/  |_.__/ \\___|  \\__,_| |_|  |_|_|_|_|_|_|\\___/|_| |_|\\__,_|_|_|_|  \\___|  (_)  \n\n");

        options();

        string Uinput = Console.ReadLine();

        if (Uinput != "")

        {

          switch (Uinput)

          {

            case "1":

              Console.Clear();

              Contestants(People); //call Contestants method

              break;

            case "2":

              Console.Clear();

              Edit(People); //call edit method

              break;

            case "3":

              Console.Clear();

              GenerateFinalists(People, Finalists); //call generating method

              break;

            case "4":

              Console.Clear();

              Instructions(); //calls instructions method

              break;

            case "5":

              Console.Clear();

              game(People, Finalists, Chosen, answers, rewards); //calls game

              break;

            case "0":

              Console.Clear();
```

```
                Console.WriteLine("\nThank you for playing"); //exits program
                Thread.Sleep(2500);
                menuLoop = false;
                Environment.Exit(-1);
                break;
            default:
                Console.Write("\n\n\t\t\t\t\t\t  Please enter a valid option"); //if invalid input is
inputted
                Thread.Sleep(1000);
                Console.Clear();
                break;
          }
        }
      }
    }
    public static void read(names[] People)
    {
      StreamReader sr = new StreamReader(@"millionaire.txt"); //reads millionare txt file
      int count = 0;
      while (!sr.EndOfStream) //Gets a value that indicates whether the current stream position is
at the end of the stream.
      {
        People[count].fname = sr.ReadLine();
        People[count].lname = sr.ReadLine();
        People[count].inter = sr.ReadLine();
        count++; //adds to count and stops at 30 resulting that code above will expecting 30 inputs
      }
      sr.Close(); //closes the streamreader
    }
    public static void readingquestions(questionare[] answers)
    {
      StreamReader sr = new StreamReader(@"ques.txt"); //reads ques txt file located in debug file
```

```
        for (int i = 0; i < 17; i++) //runs from 0 to 17 indicating that only given variables below will only be read up to 17 times
        {
            answers[i].des1 = sr.ReadLine(); //reads line one and places into answers slot i (0)
            answers[i].questions = sr.ReadLine();  //(1)
            answers[i].a = sr.ReadLine();
            answers[i].b = sr.ReadLine();
            answers[i].c = sr.ReadLine();
            answers[i].d = sr.ReadLine();
            answers[i].ans = sr.ReadLine();
            answers[i].des2 = sr.ReadLine();
        }
        sr.Close(); //closes the streamreader
    }
    public static void readingmoney(money[] rewards)
    {
        StreamReader sr = new StreamReader(@"money.txt"); //reads money txt file located in the debug file
        int count = 0;
        while (!sr.EndOfStream) //Gets a value that indicates whether the current stream position is at the end of the stream.
        {
            rewards[count].reward = sr.ReadLine(); //reads money and places in count slot(0)...
            count++; //places reward into the slot number of the array
        }
        sr.Close(); //closes the streamreader
    }
    public static void sort(names[] People) //sorts by last name by alphabetic order
    {
        for (int i = 0; i < People.Length - 1; i++)
        {
```

```
            for (int pos = 0; pos < People.Length - 1; pos++) //pos intianted by 0,then pos < People length -1
            {
               if (People[pos + 1].lname.CompareTo(People[pos].lname) < 0)
               {
                  names temp = People[pos + 1]; //temp file assigned to people[pos+1]
                  People[pos + 1] = People[pos]; //assigns people to the next array slot
                  People[pos] = temp;
               }
            }
         }
      }
      public static void display(names[] People)
      {
         for (int i = 0; i < People.Length; i++)
         {
            Console.Write($" {People[i].lname.PadRight(30)}"); //displays person and pads it 30 spaces to the right
            Console.Write($"{People[i].fname.PadRight(20)}");
            Console.WriteLine($"{People[i].inter}");
         }
      }
      public static void edit(names[] People)
      {
         bool found = false; //found boolean set to false used for checking if person is in the list
         while (found == false)
         {
            Console.Clear();
            Console.WriteLine("");
            display(People); //calls display
            Console.WriteLine("\n Who's interests would you like to change?\n"); //asks user who they want to edit
```

```
            Console.Write(":");
            string edit = Console.ReadLine(); //reads user's input
            for (int i = 0; i < People.Length; i++)
            {
               if (People[i].lname == edit)  //if the person is found will result in the code being run
               {
                  Console.WriteLine(" What would you like to change it to?:");
                  People[i].inter = Console.ReadLine();
                  found = true; //if person is found the boolean will become true
               }
            }
            if (found == false) //if boolean is flase meaning person is not found it'll print out user is not
found
            {
               Console.WriteLine(" Person not found");
               Console.ReadLine();
            }
            Console.Clear();
         }
      }
      public static void display2(names[] People)
      {
         for (int i = 0; i < People.Length; i++) //for loop that runs from 0 to the length of People
         {
            Console.Write($" {People[i].lname.PadRight(30)}");
            Console.Write($"{People[i].fname.PadRight(20)}");
            Console.WriteLine($"{People[i].inter}");
         }
      }
      public static void options()
      {
```

```
    Console.WriteLine("");                              //Options presented to users
    Console.WriteLine("\t\t\t\t\t\t   1)  Contestants\n");
    Console.WriteLine("\t\t\t\t\t\t   2)  Edit Contestants");
    Console.WriteLine("");
    Console.WriteLine("\t\t\t\t\t\t   3)  Generate Finalists");
    Console.WriteLine("");
    Console.WriteLine("\t\t\t\t\t\t   4)  Instructions");
    Console.WriteLine("");
    Console.WriteLine("\t\t\t\t\t\t   5)  Play Game");
    Console.WriteLine("");
    Console.WriteLine("\t\t\t\t\t\t   0)  Exit Game");
    Console.Write("\n\t\t\t\t\t\t  :");
}
public static void Top10(names[] People)
{
    Random Rand = new Random();
    int[] finalists = new int[10]; //finalist array 10 slots
    for (int i = 0; i < finalists.Length; i++) //for loop that runs from 0 to the length of finalists
    {
        finalists[i] = -1;  //populates all the index's with -1
    }
    int lottery;
    for (int i = 0; i < finalists.Length; i++)
    {
        lottery = Rand.Next(0, 30); //generate a random number from 0-30
        while (finalists.Contains(lottery))
        {
            lottery = Rand.Next(0, 30);
        }
        finalists[i] = lottery; //assigns random number to an index
    }
```

```
        for (int i = 0; i < finalists.Length; i++)
        {
            Console.WriteLine("");
            Console.WriteLine($"\t{People[finalists[i]].fname} {People[finalists[i]].lname} "); //merges people array into the finalists slot [i]
        }
    }
    public static void Top1(names[] People, string[] Finalists)
    {
        Random Rand = new Random();
        int[] finalists = new int[1]; //array 1 slot long used to store top1 finalist
        for (int i = 0; i < finalists.Length; i++) // starts from 0 and ends with teh finalist's size
        {
            finalists[i] = -1; //makes sure duplication doesn't take place
        }
        int lottery; //int called lottery used for fianlsits for loop
        for (int i = 0; i < finalists.Length; i++)
        {
            lottery = Rand.Next(0, 30); //generates a random number from 0,30
            while (finalists.Contains(lottery))
            {
                lottery = Rand.Next(0, 30);  //regenerates another random number from 0,30
            }
            finalists[i] = lottery; //assignsd random number to every index
        }
        for (int i = 0; i < finalists.Length; i++)
        {
            Console.WriteLine($"\t{People[finalists[i]].fname} {People[finalists[i]].lname} "); // prints top 1 finalist
        }
    }
    public static void Contestants(names[] People)
```

```
    {
        Console.WriteLine("\n These are all your contestants\n");
        display(People); //calls display method
        Console.WriteLine("\n Press Enter to Exit");
        Console.ReadLine();
    }
    public static void Edit(names[] People)
    {
        Console.WriteLine(" Edit Contestants");
        display(People); //calls display method
        edit(People); //calls edit method
        display2(People); //calls display2 method
        Console.WriteLine("\n Press Enter to Exit");
        Console.ReadLine();
    }
    public static void Instructions()
    {

Console.WriteLine("**************************************************************
************************"); //Rules
        Console.WriteLine("*\tThe rules for this game are simple                                    *");
        Console.WriteLine("*\tTo earn the million dollars you must answer 16 questions.
*");
        Console.WriteLine("*\tThe game is not case sensitive therefore any input from the user is
accepted    *");
        Console.WriteLine("*\tGoodluck                                                             *");

Console.WriteLine("**************************************************************
************************");
        Console.ReadLine();
    }
    public static void GenerateFinalists(names[] People, string[] Finalists)
    {
```

```
        Console.WriteLine("\n\tYour top 10 finalists are");
        Top10(People); //calls top10 method
        Thread.Sleep(2500);
        Console.Clear();
        Console.WriteLine("\nAnd you finalist is \n");
        Top1(People, Finalists); //calls top1 method
        Thread.Sleep(2500);
        Console.Clear();
        Console.WriteLine("Press Enter to Exit");
        Console.ReadLine();
    }

    public static void game(names[] People, string[] Finalists, string[] Chosen, questionare[] answers, money[] rewards)
    {
        int count = 0;
        for (int i = 0; i < answers.Length; i++) //loops from 0 to the length of answers
        {
            Thread.Sleep(100);
            string Uinput;
            Console.Clear();
            Console.WriteLine($" {answers[count].des1}");
            Console.WriteLine($" {answers[count].questions}");
            Console.WriteLine($" {answers[count].a}");
            Console.WriteLine($" {answers[count].b}");
            Console.WriteLine($" {answers[count].c}");
            Console.WriteLine($" {answers[count].d}");
            Console.WriteLine($" {answers[count].des1}");

            Console.Write("\n  :");
            Uinput = Console.ReadLine().ToUpper(); //converts user input into uppercase
```

```
            if (Uinput == answers[count].ans) //if Uninput contains correct answer the following will be
displayed
            {
               Console.WriteLine($"\n {answers[count].des1}");
               Console.WriteLine($"\t\t\t\t\t Correct you have just won ${rewards[i].reward}");
               Console.WriteLine($" {answers[count].des2}\n");
               Thread.Sleep(2500);
               count++; //
            }
            else
            {
               Console.WriteLine($" {answers[count].des1}");
               Console.WriteLine("  That is the wrong answer");
               Console.WriteLine($"  The correct answer was {answers[count].ans}");
               Console.WriteLine($"  You are going home with $ {rewards[i - 1].reward}");
               Console.WriteLine($" {answers[count].des2}\n");
               Thread.Sleep(3000);
               menu(People, Finalists, Chosen, answers, rewards); //retuns user back to the menu
            }
         }
      }
   }
}
```