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

namespace CitECycling
{
   class Program
   {

      // Main function
      static void Main(string[] args)
      {

         // This is my way of repeating the menu, when the menu equals 1, it will repeat, if it equals anything else, it will not repeat.
         int iCorrectOption = 1;

         // if iCorrectOption equals 1, continue to repeat.
         while (iCorrectOption == 1)
         {

            // Call the menu function everytime we want to repeat
            int iOption = getOption();

            //Switch allows us to build a menu (easier if statement)
            switch (iOption)
            {
```

```
// If the person selects menu option 1
case 1:

   // Call the "display all locations" function with no arguments
   displayAllLocations();

   // Once it's finished calling, re-loop the menu (break exits the switch statement
   //but will still allow for the outside while loop to function).
   break;

// If the person selects menu option 2
case 2:

   // Call the "add participants" function with no arguments
   addParticipants();

   // Once it's finished calling, re-loop the menu (break exits the switch statement
   //but will still allow for the outside while loop to function).
   break;

case 3:

   // Call the "Display registered participants" function with no arguments
   displayRegisteredParticipants();

   // Once it's finished calling, re-loop the menu (break exits the switch statement
   //but will still allow for the outside while loop to function).
   break;

case 4:
```

```
                // Say goodbye, and dont call anything (since there's nothing left, it'll break and close program.
                Console.WriteLine("Closing Program, goodbye!");

                // Prevent the menu from re-looping.
                iCorrectOption--;

                // Exit
                break;

            }

        }

    }

    // Menu options and validation function
    static int getOption()
    {
        // Declare variable to store user input
        int iOption = -1;

        // Display user options
        Console.WriteLine("Please enter your option: ");
        Console.WriteLine("1. View Cit-E Cycling Locations ");
        Console.WriteLine("2. Register to participate ");
        Console.WriteLine("3. Display all registered participants ");
        Console.WriteLine("4. Exit ");

        // Get user input
        iOption = Convert.ToInt32(Console.ReadLine());
```

```
    // Validate user input
    // It must be within the correct range
    while (iOption < 1 || iOption > 4)
    {
        Console.WriteLine("Please enter a valid option between 1 and 4");
        iOption = Convert.ToInt32(Console.ReadLine());
    }

    // return validated option
    return iOption;
}

// Displaying all locations function
static void displayAllLocations()
{

    // Defines a new streamreader instance that will look through locations.txt
    using (StreamReader sr = new StreamReader("locations.txt"))
    {

        // String slocation will store the location
        string sLocation = "";

        string sDate = "";
        // String sdate will store the date of a location

        string sLine = "";
        // String sline will store the text of the current line

        int iLineNumber = 0;
```

```
        // This integer will store the current line number

        // Read and display lines from the file until the end of
        // the file is reached.
        while ((sLine = sr.ReadLine()) != null)
        {

            // Add 1 to the current linenumber (used to check if a line is odd/even)
            iLineNumber++;

            // If the line number is odd, set slocation to current line value
            if ((iLineNumber % 2) == 1)
            {
                sLocation = sLine;
            }

            // If the line number is even, set sdate to current line value and print location and date
            if ((iLineNumber % 2) == 0)
            {
                sDate = sLine;
                Console.WriteLine("Location: " + sLocation + " on date: " + sDate);
            }
        }
    }

    Console.WriteLine(" ");
    Console.WriteLine("Press any key to return to main menu!");

    // Awaits user key input then sends back to caller
    Console.ReadKey();
}
```

```
// This function will check locations to ensure they're valid/in locations.txt
// The function requires a location as an argument, for example "London" and will check if that location exists and return with a 0/1 (false/true)
static int checkLocation(string sLocation)
{

    // sLine will be replaced with contents of current line on every loop
    String sLine = "";

    // Defines a new streamreader instance that will look through locations.txt
    using (StreamReader sr = new StreamReader("locations.txt"))
    {

        // Loops and ensure's the line is not empty
        while ((sLine = sr.ReadLine()) != null)
        {

            // Checks if current line is the location we're looking for (sLocation being the argument of the function)
            if (sLine == sLocation)
            {

                // If it's found in this line, it'll return a 1 (true - location exists) to the caller.
                return 1;

            }

        }

        // If it's not found in this line, it'll ask the person again.
```

```
            return 0;
        }

    }

    // This function will check timeslots to ensure they're 1/2/3
    // The function requires a timeslot as an argument, for example "London" and will check if that location exists and return with a 0/1 (false/true)
    static int checkTimeslot(int iTimeSlot)
    {

        // If timeslot is not between 1-3, return a false
        if (iTimeSlot < 1 || iTimeSlot > 3)
        {
            return 0;
        }

        // If timeslot is between 1-3, return a true
        return 1;

    }

    // This function allows someone to add participants to the event.
    static void addParticipants()
    {

        // Declare variabe and set it to -1
        int iParticipants = -1;

        // Validitiy check variable, used for a few statements to check inputs
        int iInvalid = 0;
```

```
// Location variable, lets us assign which location we're adding people to
String sLocation = "";

// While the location isn't valid, loop the section below
while (iInvalid == 0)
{

   // Ask the person where they're singing up for
   Console.WriteLine("Please enter which location you'd like to sign people up for!");

   // Set sLocation to their response
   sLocation = Console.ReadLine();

   // Run the validation on the location to check it exists
   int iValidLocation = checkLocation(sLocation);

   // If the location is invalid
   if (iValidLocation == 0)
   {
      Console.WriteLine("Invalid location! Please enter a valid location.");
   }

   // If the location is valid
   else
   {

      // Exit current loop
      iInvalid++;
   }
}
```

```
// This will define the text file that data will be placed into.
string sFilename = sLocation + ".txt";

// Put iInvalid back to 0 (to be re-used)
iInvalid = 0;

// Ask how many people they're registering
Console.WriteLine("How many people are you registering?");

// Convert text input into an integer
iParticipants = Convert.ToInt32(Console.ReadLine());

// For every participant, add 1 to times looped. If there's nobody left, stop asking
for (int i = 0; i < iParticipants; i++)
{


   // Define sname for this person and make it so it's blank
   string sName = "";

   // Define iNumber for this person and make it so it's -1 (will be used for validation)
   // Had to change to a "long" because it was throwing overflow exceptions.
   long iNumber = -1;

   // Define iSlot, this will be used to determine what slot the person will be in.
   int iSlot = -1;

   // Ask name of person
   Console.WriteLine("What's the name of person #" + i + "?");
```

```
// save rsponse in sName
sName = Console.ReadLine();

// This will run to ensure length of phone number is 11
while (iNumber == -1)
{

   // Ask for the contact number of person
   Console.WriteLine("What's the contact number of person #" + i + "?");

   // Save number in iNumber as an integer
   iNumber = Convert.ToInt64(Console.ReadLine());

   // Fetch the length of the phone number
   iInvalid = iNumber.ToString().Length;

   // If the length is not 11
   if (iInvalid != 11)
   {
      Console.WriteLine("Invalid phone number, try again!");

      // ask the question again (throws back into loop)
      iNumber = -1;
   }

}

iInvalid = 0;

// Verify slot
while (iInvalid == 0)
```

```
    {

        // Ask which slot they're in
        Console.WriteLine("What's the slot of person #" + i + "?");

        // Convert to an integer and save in Islot
        iSlot = Convert.ToInt32(Console.ReadLine());

        // Check timeslot is valid, returns with 0/1 to represent false/true
        iInvalid = checkTimeslot(iSlot);

        if (iInvalid == 0)
        {
            Console.WriteLine("Incorrect timeslot, please enter again!");
        }

    }

    // Write this person into the appropriate text file, adding true means it wont overwrite
    using (StreamWriter sw = new StreamWriter(sFilename, true))
    {
        sw.WriteLine(sName);
        sw.WriteLine(iNumber);
        sw.WriteLine(iSlot);


    }


}
```

```
    Console.WriteLine(" ");
    Console.WriteLine("Press any key to return to main menu!");

    // Awaits user key input then sends back to caller
    Console.ReadKey();

}



// This will display all registered participants for an event
static void displayRegisteredParticipants()
{

    // This will be the location we're checking
    string sLocation = "";

    // This is a validation check variable
    int iInvalid = 0;

    // This is the line when the streamreader is looping through it
    string sLine = "";

    // This is the current line number
    int iLineNumber = 0;

    // This is a full list of participants (couldn't get an array to add values)
    List<string> lParticipants = new List<string>();
    List<string> lContacts = new List<string>();
    List<string> lSlotone = new List<string>();
    List<string> lSlottwo = new List<string>();
```

```
List<string> lSlotthree = new List<string>();

// If the answer given is invalid/default, repeat
while (iInvalid == 0)
{

   // Ask the location
   Console.WriteLine("Please enter which location you'd like to view data for!");

   // Read the answer and put it in sLocation
   sLocation = Console.ReadLine();

   // Check the location to see if it's in locations.txt
   int iValidLocation = checkLocation(sLocation);

   // If it's invalid, print and keep us in loop
   if (iValidLocation == 0)
   {
      Console.WriteLine("Invalid location! Please enter a valid location.");
   }

   // If it's valid, exit the loop by setting iInvalid to 1
   else
   {

      // Exit current loop
      iInvalid++;
   }
}

// Setting text file for current locartion
```

```
string sFilename = sLocation + ".txt";

// New streamreader instance to read the given text file
using (StreamReader sr = new StreamReader(sFilename))
{

   // Loops and ensure's the line is not empty
   while ((sLine = sr.ReadLine()) != null)
   {

      // Add 1 to the current linenumber (used to check if a line line 1,2,3 in the format)
      iLineNumber++;

      // If the line number is 1, add the participant's name to the list
      if ((iLineNumber % 3) == 1)
      {
         lParticipants.Add(sLine);
      }

      // If the line number is 2, add tge participants contact to the list
      if ((iLineNumber % 3) == 2)
      {
         lContacts.Add(sLine);
      }

      // If the line number is 3, add 1 person to the appropriate slot.
      if ((iLineNumber % 3) == 0)
      {
         // if slot is one, add someone to that slot
         if (sLine == "1")
         {
```

```
            lSlotone.Add(sLine);
        }

        // if slot is two, add someone to that slot
        if (sLine == "2")
        {
            lSlottwo.Add(sLine);
        }

        // if slot is three, add someone to that slot
        if (sLine == "3")
        {
            lSlotthree.Add(sLine);
        }
    }

}

// information printout section
Console.WriteLine();
Console.WriteLine(sLocation + " Event information");
Console.WriteLine("Total amount of participants: " + lParticipants.Count);
Console.WriteLine("Participants in slot 1: " + lSlotone.Count);
Console.WriteLine("Participants in slot 2: " + lSlottwo.Count);
Console.WriteLine("Participants in slot 3: " + lSlotthree.Count);
Console.WriteLine();

// set iParticipantnumber to a default of 1
int iParticipantNumber = 1;

// loops people in list
```

```
            foreach (var sParticipant in lParticipants)
            {
                Console.WriteLine("Participant " + iParticipantNumber + " name: " + sParticipant);
                iParticipantNumber++;
            }

            Console.WriteLine(" ");
            Console.WriteLine("Press any key to return to main menu!");

            // Awaits user key input then sends back to caller
            Console.ReadKey();
        }


    }
  }

}
```