```csharp
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.IO;
using System.Data;
using System.Text.RegularExpressions;
using System.Windows;


namespace week7_LibraryRevision_1
{
    class Program
    {
        //VARIABLES

        //static variables here
        static string studentID; //holds value of student ID durring processes
        static string currentName; //holds value of student's name durring processes
        static string resourcesCheckedOut; //how many resources student currently has checked out
        static string studentResource1; //holds value of resource student has checked out
        static string studentResource2; //holds value of resource student has checked out
        static string studentResource3; //holds value of resource student has checked out
        static string seperator = ",";
        static string currentTitleSearch; //holds value of resource during check out process
        static string currentReturnResourceTitle; //holds value of resource during return process
        static string[] currentStudentArray = new string[6]; //TODO CAN BE A LIST?
        static StringBuilder currentStudentHeaderBuilder = new StringBuilder();

        //Declare Dictionaries and Lists
        static Dictionary<string, Int16> staticCatalog = new Dictionary<string, Int16>(StringComparer.OrdinalIgnoreCase); //Declaring fixed dictionary
        static Dictionary<string, string> staticIDCatalog = new Dictionary<string, string>(StringComparer.OrdinalIgnoreCase); //Declaring fixed dictionary
        static Dictionary<string, Int16> workingCatalog = new Dictionary<string, Int16>(StringComparer.OrdinalIgnoreCase); //Declaring mutable catalog dictionary
        static Dictionary<string, string> studentRoster = new Dictionary<string, string>(StringComparer.OrdinalIgnoreCase); //Declaring fixed catalog dictionary
        static List<string> resourcesOutList = new List<string>();


        //METHODS

        //MAIN MENU
        static void MenuDisplay()
                {
                    //PRINT MENU TO SCREEN
                    Console.Clear();
                    Console.WriteLine("*************** Bootcamp Resources Checkout System ****************\n");

                    string[] optionsMenu = new string[] { "1 - View Students", "2 - View Available Resources", "3 - Resources Checked Out", "4 - View Student Account", "5 - Check Out Item", "6 - Return Item", "7 - Exit" };

                    Console.WriteLine();
                    for (int i = 0; i < optionsMenu.Length; i++)
                    {

                        Console.WriteLine(optionsMenu[i]);
                    }
                    Console.Write("\nChoose a menu item: ");

                    string input = Console.ReadLine();
                    input = input.Trim();

                    //int menuChoice;
                    int menuChoice;
                    bool res = int.TryParse(input, out menuChoice);

                    if (res == false)
                    {
                        Console.Clear();
                        Console.WriteLine("Enter a number from 1 to 7 to make a selection");
                        MenuDisplay();
                    }


                    else if (menuChoice < 1 || menuChoice > 7)
                    {
                        Console.Clear();
                        Console.WriteLine("*************** Bootcamp Resources Checkout System ****************\n\n");

                        Console.WriteLine("Enter a number from 1 to 6 to make a selection");
                        MenuDisplay();
                    }


                    switch (menuChoice)
                    {
                        case 1:
                            ListAllStudentsAlphabetical();
                            break;
                        case 2:
                            ListAvailableResourcesAlphabetical2();
                            break;
                        case 3:
                            ResroucesOutWithStudentName();
                            break;
                        case 4:
                        {
                        VerifyID2();
                            StudentProfile();
                            break;
                        }
                        case 5:
                        {

                        VerifyID2();
                            StudentCheckOut();
                            break;
                        }
                        case 6:
                        {
                            VerifyID2();
                            ResourceReturn();
                            break;
                        }

                        case 7:
                            Exit();
                            break;
                        default:

                            break;
                    }

                }

        //VERIFY STUDENT ID...validate student ID. Use with check out/return 
          process, view student account process
        static void VerifyID2() //search for student record. If it exists, sets 
          Student ID Variable. Use for all operations that require student ID
        {


            string choice;
            string pattern = @"^\d{3}$";
            Regex matchInput = new Regex(pattern, RegexOptions.IgnoreCase);
            do
            {
                Console.Clear();
                Console.WriteLine("*************** Bootcamp Resources Checkout 
                  System ***************\n\n");
                Console.Write("Enter a 3 digit Student ID or M to return to Main 
                   Menu: ");
                choice = Console.ReadLine().ToUpper();

                Match m = matchInput.Match(choice);
                if (m.Success)
                {
                    if (!File.Exists(choice + ".txt"))
                    {
                        Console.WriteLine("That student is not in our records");
                        Console.Clear();
                        MenuDisplay();
                    }
                    studentID = choice;
                    break;

                }
                else if (choice == "M")
                {
                    Console.Clear();
                    MenuDisplay();
                }

                else
                {

                    choice = "repeat";
                    Console.Clear();
                }
            } while (choice == "repeat");

            Console.Write("\nEnter Student ID: ");
            //string input = Console.ReadLine();

            // check for student in records




            //read each line of a text file and assign to variables
            StreamReader sr = new StreamReader(studentID + ".txt"); //studentID 
              assigned when user enters
            using (sr)
            {
                string line;
                int counter = 0;

                //assign values from student text file to currentStudent Array
                while (counter < 6)
                {
                    line = sr.ReadLine();
                    currentStudentArray[counter] = line;
                    counter++;
                }

                //assign index values from currentStudent Array to individual variables
                studentID = currentStudentArray[0].ToString();
                currentName = currentStudentArray[1].ToString();
                resourcesCheckedOut = currentStudentArray[2].ToString();
                studentResource1 = currentStudentArray[3].ToString();
                studentResource2 = currentStudentArray[4].ToString();
                studentResource3 = currentStudentArray[5].ToString();
            }

        }

        //CURRENT STUDENT HEADER
        static void CurrentStudentHeader()
        {
            StringBuilder currentStudentHeaderBuilder = new StringBuilder();
            Console.WriteLine(currentStudentHeaderBuilder.Append("Current Student: " + currentName));
            Console.WriteLine();
        }

        //CHECK OUT PROCESSES
        static void StudentCheckOut()
        {

            Console.Clear();
            Console.WriteLine("*************** Bootcamp Resources Checkout System ***************\n\n");
            CurrentStudentHeader();


            //check resourcesCheckedOut variable...if >2, student may not check out books, return to menu
            if (int.Parse(resourcesCheckedOut) > 2)
            {

                StringBuilder maxResourcesOut = new StringBuilder();
                maxResourcesOut.Append(currentName).Append(" has checked out the maximum number of resources.");
                Console.WriteLine(maxResourcesOut);

                Console.Write("\nPress any key to return to Main Menu");
                Console.ReadKey();
                MenuDisplay();
            }

            TitleSearchAvailabilityAlt(); //search for title availability

            MenuDisplay();


        }
        static void TitleSearchAvailabilityAlt()
        {

            //Console.WriteLine("Enter resource ID of item to checkout or V to view resource IDs");
            //string userInput = Console.ReadLine();

            string choice = null;

            do
            {
                Console.Clear();
                Console.WriteLine("*************** Bootcamp Resources Checkout System ****************\n\n");
                CurrentStudentHeader();

                Console.Write("Enter resource ID of item to checkout or V to view resource IDs: ");
                choice = Console.ReadLine().ToUpper().Trim();
                if (choice == "V")
                {
                    Console.Clear();
                    Console.WriteLine("*************** Bootcamp Resources Checkout System ****************\n\n");
                    CurrentStudentHeader();

                    ListResourceWithID();
                    Console.Write("\nEnter resource ID of item to checkout or M for Main Menu: ");
                    choice = Console.ReadLine().ToUpper();
                    if (choice == "M")
                    {
                        Console.Clear();
                        MenuDisplay();
                    }
                    else if (staticIDCatalog.ContainsKey(choice))
                    {
                        break;
                    }
                    else
                    {
                        choice = "repeat";
                        Console.Clear();
                    }

                }
                else if (staticIDCatalog.ContainsKey(choice))
                {

                    break;
                }

                else
                {

                    choice = "repeat";
                    Console.Clear();
                }
            } while (choice == "repeat");

            currentTitleSearch = staticIDCatalog[choice];

            //check to see if there is a copy available to check out.
            if (workingCatalog[currentTitleSearch] == 0)
            {
                Console.Clear();
                Console.WriteLine("There are no copies of " + currentTitleSearch ↩
                   +" availalbe at this time\n");
                //turn this do loop into a method?
                //string choice;
                do
                {
                    Console.WriteLine("Enter S to search again, or M to return   ↩
                    to the Main Menu\n: ");
                    choice = Console.ReadLine().ToUpper();
                    if (choice == "M")
                    {
                        Console.Clear();
                        MenuDisplay();
                    }
                    else if (choice == "S")
                    {
                        TitleSearchAvailabilityAlt();
                    }

                    else
                    {

                        choice = "repeat";
                        Console.Clear();
                    }
                } while (choice == "repeat");
            }

             // TODO CONFIRM STUDENT WANTS TO CHECKOUT HERE?

            //decrement resource availability in workingCatalog dictionary
            if (workingCatalog.ContainsKey(currentTitleSearch)) //WTF
            {
                workingCatalog[currentTitleSearch] -= 1;

                //save working Catalog to File
                SaveWorkingCatalogToFile();

                //save resource to student file and write to ResourcesOutList
                SaveResourceToStudentFile();
                SaveToResourcesOutList();
                Console.Clear();
                Console.WriteLine("*************** Bootcamp Resources Checkout System ***************\n\n");
                CurrentStudentHeader();

                Console.WriteLine(currentName + " has checked out " + currentTitleSearch + "\n"); // TODO I would like this title to come from the Array for correct formatting

                if (int.Parse(resourcesCheckedOut) > 2)
                {
                    Console.Write("Press any key to return to Main Menu");
                    Console.ReadKey();
                    Console.Clear();
                    MenuDisplay();
                }

                //offer option to check out again..turn into method?
                //string choice;
                do
                {
                    Console.WriteLine("\nEnter S check out another item, or M to return to the Main Menu: \n");
                    choice = Console.ReadLine().ToUpper();
                    if (choice == "M")
                    {
                        Console.Clear();
                        MenuDisplay();
                    }
                    else if (choice == "S")
                    {

                        TitleSearchAvailabilityAlt();
                    }

                    else
                    {

                        choice = "repeat";
                        Console.Clear();
                    }
                } while (choice == "repeat");

            }
        }
        static void SaveResourceToStudentFile()
        {

            //find resource in currentStudent array
            for (int i = 3; i < currentStudentArray.Length; i++)
            {
                if ((currentStudentArray[i] != "-")) //extra parenthesis?  TODO
                {
                    continue;
                }

                if ((currentStudentArray[i] == "-")) //extra parenthesis?  TODO
                {
                    //assign currentTitleSearch value to array
                    //currentStudent[i] = currentTitleSearch;
                    //assign currentTitleSearch value to currentStudent list
                    if (i == 3)
                    {
                        currentStudentArray[i] = currentTitleSearch;
                        studentResource1 = currentTitleSearch;

                    }
                    else if (i == 4)
                    {

                        currentStudentArray[i] = currentTitleSearch;
                        studentResource2 = currentTitleSearch;
                    }
                    else if (i == 5)
                    {

                        currentStudentArray[i] = currentTitleSearch;
                        studentResource3 = currentTitleSearch;
                    }

                    //math to increase number of books checked out
                    int x = int.Parse(resourcesCheckedOut);
                    x++;
                    string y = x.ToString();
                    currentStudentArray[2] = y;

                    //update number of resources student has checked out
                    resourcesCheckedOut = y;

                    break;
                }

            }
            //write updated student information to file
```

```
                    using (StreamWriter SaveStudentFile = new StreamWriter(studentID + ".txt")) //delete student text file
                    {

                    }

                    using (StreamWriter sw = File.AppendText(studentID + ".txt")) //write new values to student text tile
                    {
                        sw.WriteLine(studentID);
                        sw.WriteLine(currentName);
                        sw.WriteLine(resourcesCheckedOut);
                        sw.WriteLine(studentResource1);
                        sw.WriteLine(studentResource2);
                        sw.WriteLine(studentResource3);
                    }

        }

        //WRITE RESOURCE TO "RESOURCESOUT" LIST
        static void SaveToResourcesOutList()
        {

            //use stringbuilder to concat studentName and currentTitleSearch
            StringBuilder resourceAndStudentCSV = new StringBuilder();
            resourceAndStudentCSV.Append(currentTitleSearch).Append(seperator).Append(currentName);
            string resourceAndStudent = resourceAndStudentCSV.ToString();

            //add currently checked out resource to resourcesOutList
            resourcesOutList.Add(resourceAndStudent);
            StreamWriter saveResourcesOutToText = new StreamWriter("resourcesOut.txt", true);
            using (saveResourcesOutToText)
            {
                saveResourcesOutToText.WriteLine(resourceAndStudent);
            }


        }

        //SAVE workingCatalog TO FILE (after checkout or return..updates resources checked out/available)
        static void SaveWorkingCatalogToFile()
        {
            using (StreamWriter SaveWorkingCatatlog = new StreamWriter("working-catalog.txt"))
            {
                foreach (KeyValuePair<string, Int16> kvp in workingCatalog)
                {
                    StringBuilder workingCatalogBuildString = new StringBuilder();
                    string saveWorkingCatalog = (workingCatalogBuildString.Append(kvp.Key).Append(seperator).Append(kvp.Value)).ToString();
                    SaveWorkingCatatlog.WriteLine(saveWorkingCatalog);
                }
            }
        }

        //RETURN PROCESS
        static void ResourceReturn() //takes student ID as argument
        {

            //does student have any resources out?

            if ((int.Parse(resourcesCheckedOut) < 1 ))
            {
                Console.Clear();
                Console.WriteLine("*************** Bootcamp Resources Checkout System ***************\n\n");
                CurrentStudentHeader();
                Console.WriteLine(currentName + " has 0 resrouces checked out.");
                Console.Write("\nPress any key to return to Main Menu");
                Console.ReadKey();
                Console.Clear();
                MenuDisplay();
            }

            CurrentStudentResourcesCheckedOut();
            ReturnResourcetoWorkingCatalog();
            ReturnResourceToStudentFile();
            RemoveResourceFromResourceOutList();
            Console.Clear();
            Console.WriteLine("*************** Bootcamp Resources Checkout System ***************\n\n");
            CurrentStudentHeader();

            StringBuilder hasReturned = new StringBuilder();
            hasReturned.Append(currentName).Append(" has returned ").Append(currentReturnResourceTitle);
            Console.WriteLine(hasReturned);


            string choice;
            do
            {
                Console.WriteLine("\nEnter R to return another item for " + currentName + " or M to return to the Main Menu\n");
                choice = Console.ReadLine().ToUpper();
                if (choice == "R")
                {
                    Console.Clear();
                    ResourceReturn();
                }
                else if (choice == "M")
                {
                    MenuDisplay();
                }

                else
                {

                    choice = "repeat";
                    Console.Clear();
                }
            } while (choice == "repeat");


            MenuDisplay();
            ;

        } //what is this?
        static void CurrentStudentResourcesCheckedOut()
        {
            Console.Clear();

            Dictionary<string, string> returnOptions = new Dictionary<string,
              string>();
            int counter = 1;
            for (int i = 3; i < currentStudentArray.Length; i++)
            {
                if ((currentStudentArray[i] == "-")) //does this have an extra
                  parenthesis?  TODO
                {
                    continue;
                }
                if ((currentStudentArray[i] != "-")) //extra parenthesis?  TODO
                {
                        returnOptions.Add(counter.ToString(),
                    currentStudentArray[i]);
                        counter++;
                }

            }

            //chooose which item to return

            string input;
            do
            {
                Console.Clear();
                Console.WriteLine("*************** Bootcamp Resources Checkout
                  System ***************\n\n");
                CurrentStudentHeader();
                Console.WriteLine("Resources checked out:\n");
                foreach (KeyValuePair<string,string> kvp in returnOptions)
                {
                    Console.WriteLine(kvp.Key + ".  " + kvp.Value);
                }


                Console.Write("\n\nEnter the number of the item you would like to return: ");
                input = Console.ReadLine().Trim();
                if (returnOptions.ContainsKey(input))
                {
                    string value;
                    if (returnOptions.TryGetValue(input, out value))
                    {
                        currentReturnResourceTitle = value;
                        Console.WriteLine("you chose to return: " + currentReturnResourceTitle);    //DEGBUG
                    }
                }
            } while (!(returnOptions.ContainsKey(input)));

        }
        static void ReturnResourcetoWorkingCatalog() //saves returned resource to txt file
        {
            Console.Clear();
            Console.WriteLine("************** Bootcamp Resources Checkout System ***************\n\n");
            CurrentStudentHeader();

            //This increments available parameter in workingCatalog dictionary
            if (workingCatalog.ContainsKey(currentReturnResourceTitle)) //WTF
            {
                workingCatalog[currentReturnResourceTitle] ++;
                Console.WriteLine(currentName + " has returned " + currentReturnResourceTitle + ".");

                //save working Catalog to File
                SaveWorkingCatalogToFile();
            }
        }
        static void ReturnResourceToStudentFile() //saves returned resource to student txt file, updates currentStudent Array
        {

            //find resource in currentStudent array
            for (int i = 3; i < currentStudentArray.Length; i++)
            {
                if ((currentStudentArray[i] != currentReturnResourceTitle)) // extra parenthesis?  TODO
                {
```

```csharp
                        continue;
                    }


                    if ((currentStudentArray[i] == currentReturnResourceTitle)) // extra parenthesis?  TODO
                    {
                        //reset array resource value to "-"
                        //reset student resource variable to "-"
                        if (i == 3)
                        {
                            currentStudentArray[i] = "-";
                            studentResource1 = "-";

                        }
                        else if (i == 4)
                        {

                            currentStudentArray[i] = "-";
                            studentResource2 = "-";
                        }
                        else if (i == 5)
                        {

                            currentStudentArray[i] = "-";
                            studentResource3 = "-";
                        }

                        //math to decrease number of books checked out
                        int x = int.Parse(resourcesCheckedOut);
                        x--;
                        string y = x.ToString();
                        currentStudentArray[2] = y;

                        //update number of resources student has checked out
                        resourcesCheckedOut = y;

                        break;
                    }

                }
                //write updated student information to file
                using (StreamWriter SaveStudentFile = new StreamWriter(studentID + ".txt")) //delete student text file
                {

                }

                using (StreamWriter sw = File.AppendText(studentID + ".txt")) // write new values to student text file
                {
                    sw.WriteLine(studentID);
                    sw.WriteLine(currentName);
                    sw.WriteLine(resourcesCheckedOut);
                    sw.WriteLine(studentResource1);
                    sw.WriteLine(studentResource2);
                    sw.WriteLine(studentResource3);
                }

            }

            //REMOVE RESOURCE FROM "RESOURCESOUT" LIST
            static void RemoveResourceFromResourceOutList()
            {
                //find index of resource and student name

                string returnResourceAndStudent = currentReturnResourceTitle + seperator + currentName;

                for (int i = 0; i < resourcesOutList.Count; i++)
                {
                    if (resourcesOutList[i].ToString().Equals(returnResourceAndStudent,StringComparison.CurrentCultureIgnoreCase))
                    {
                        resourcesOutList.RemoveAt(i);
                        File.Delete("resourcesOut.txt");
                        StreamWriter updateResourcesOutTextFile = new StreamWriter("resourcesOut.txt"); //TODO FIX!! not writing to file properly
                        using (updateResourcesOutTextFile)
                        {
                            foreach (string item in resourcesOutList)
                            {
                                updateResourcesOutTextFile.WriteLine(item);
                            }
                        }

                    }


                }

            }

            //LIST STUDENTS
            static void StudentProfile()
            {
                Console.Clear();
                Console.WriteLine("*************** Bootcamp Resources Checkout System ***************\n\n");
                Console.WriteLine("\nStudent ID: " + studentID);
                Console.WriteLine("Name: " + currentName);
                Console.WriteLine("\n\n" + resourcesCheckedOut + " resources checked
                   out:\n");
                int counter = 1;
                for (int i = 3; i < currentStudentArray.Length; i++)
                {
                    if ((currentStudentArray[i] != "-")) //extra parenthesis?  TODO
                    {
                        Console.WriteLine(counter.ToString() + ". " +
                         currentStudentArray[i]);
                        counter++;

                    }

                }
                Console.Write("\nPress any key to return to Main Menu");
                Console.ReadKey();
                MenuDisplay();
            }
            static DataTable CreateStudentRosterTable(Dictionary<string, string>
              dict) //creates table from student roster dictionary, returns a table
            {
                DataTable table = new DataTable();
                table.Columns.Add("Student ID", typeof(string)); //converting a
                  dictionary to a table will always have only two columns..what if I
                   want to combine dictionaries? should I just store this all in a
                  table?
                table.Columns.Add("Student Name", typeof(string));

                foreach (KeyValuePair<string, string> kvp in dict) //adds key and
                  value of dictionary to table
                {
                    table.Rows.Add(kvp.Key, kvp.Value);
                }
                //after the for each loop, a table exists with all student Id and
                  Name in rows
                return table;
            }
            static void ListAllStudentsAlphabetical()
            {
                Console.Clear();
                Console.WriteLine("*************** Bootcamp Resources Checkout
                  System ****************\n\n");

                //DICTIONARY TO DATATABLE
                DataTable table = CreateStudentRosterTable(studentRoster); //returns
                   a table of students and id to new table

                //create dataview object of table so I can sort it
                DataView view = new DataView(table);

                //sorts dataview object by columnn named Name - ascending
                view.Sort = "Student Name ASC";

                //print columnn headers
                foreach (DataColumn column in table.Columns)
                {
                    Console.Write(column.ColumnName + "\t");
                }
                Console.WriteLine();
                Console.WriteLine();

                //print sorted data table row by row
                foreach (DataRowView row in view)
                {
                    Console.WriteLine("  {0}\t\t{1}", row[0], row[1]);
                }
                Console.WriteLine("\n");
                Console.Write("Press any key to return to Main Menu");
                Console.ReadKey();
                Console.Clear();
                MenuDisplay();
                //return dict;
            }
            static Dictionary<string, string> LoadStudentRoster(Dictionary<string, string> dict) //loads all student and id from text file to dictionary
            {
                string line;
                string[] keyAndValue;
                //List<string> students = new List<string>();
                StreamReader sr = new StreamReader(@"student-roster.txt");
                using (sr)
                {
                    while ((line = sr.ReadLine()) != null)
                    {
                        keyAndValue = line.Split(',');
                        dict.Add(keyAndValue[0], keyAndValue[1]);
                        Array.Clear(keyAndValue, 0, keyAndValue.Length);
                    }

                    return dict;
                }

            }


            //LIST AVAIALABLE RESOURCES
            static DataTable CreateAvailableResourceTable(Dictionary<string, Int16> dict) //creates table from workingCatalog dictionary, returns a table
            {
                DataTable table = new DataTable();
                table.Columns.Add("Available", typeof(Int16));
                table.Columns.Add("Resource", typeof(string)); //converting a dictionary to a table will always have only two columns..what if I want to combine dictionaries? should I just store this all in a table?

                foreach (KeyValuePair<string, Int16> kvp in dict) //adds key and 
                  value of dictionary to table
                {
                    table.Rows.Add(kvp.Value, kvp.Key);
                }
                //after the for each loop, a table exists with all student Id and 
                  Name in rows
                return table;
            }
            static void ListAvailableResourcesAlphabetical2()
            {
                Console.Clear();
                Console.WriteLine("*************** Bootcamp Resources Checkout 
                  System ****************\n\n");

                //DICTIONARY TO DATATABLE
                DataTable table = CreateAvailableResourceTable(workingCatalog); // 
                  returns a table of students and id to new table

                //create dataview object of table so I can sort it
                DataView view = new DataView(table);

                //sorts dataview object by columnn named Name - ascending
                view.Sort = "Resource ASC";

                //print columnn headers


                foreach (DataColumn column in table.Columns)
                {
                    Console.Write(column.ColumnName + "\t\t");
                }
                Console.WriteLine();
                Console.WriteLine();

                //print sorted data table row by row
                foreach (DataRowView row in view)
                {
                    Console.WriteLine("  {1}\t" + "\t{0}", row[1], row[0]);

                }
                Console.WriteLine("\n");
                Console.Write("Press any key to return to Main Menu");
                Console.ReadKey();
                Console.Clear();
                MenuDisplay();
                //return dict;
            }

            //LIST RESOURCES OUT + STUDENT NAME
            static void ResroucesOutWithStudentName()
            {
                Console.Clear();
                Console.WriteLine("Resources Checked Out:\n");
                resourcesOutList.Sort();
                for (int i = 0; i < resourcesOutList.Count; i++)
                {
                    Console.WriteLine(resourcesOutList[i]);
                }

                Console.Write("\n\nPress any key to return to Main Menu");
                Console.ReadKey();
                Console.Clear();
                MenuDisplay();
            }

            //LIST RESOURCE IDs
            static DataTable CreateResourceIDTable(Dictionary<string, string> dict) //creates table from static-ID-cataolog dictionary
            {
                DataTable table = new DataTable();
                table.Columns.Add("Resource ID", typeof(string));
                table.Columns.Add("Resource", typeof(string)); //converting a dictionary to a table will always have only two columns..what if I want to combine dictionaries? should I just store this all in a table?

                foreach (KeyValuePair<string, string> kvp in dict) //adds key and value of dictionary to table
                {
                    table.Rows.Add(kvp.Key, kvp.Value);
                }
                //after the for each loop, a table exists with all student Id and Name in rows
                return table;
            }
            static void ListResourceWithID() //sorts alphabetical and displays
            {
                Console.Clear();
                Console.WriteLine("*************** Bootcamp Resources Checkout System ****************\n\n");

                //DICTIONARY TO DATATABLE
                DataTable table = CreateResourceIDTable(staticIDCatalog); //returns a table of students and id to new table

                //create dataview object of table so I can sort it
                DataView view = new DataView(table);

                //sorts dataview object by columnn named Name - ascending
                view.Sort = "Resource ASC";

                //print columnn headers


                foreach (DataColumn column in table.Columns)
                {
                    Console.Write(column.ColumnName + "\t\t");
                }
                Console.WriteLine();
                Console.WriteLine();

                //print sorted data table row by row
                foreach (DataRowView row in view)
                {
                    Console.WriteLine("  {1}\t" + "\t{0}", row[1], row[0]);

                }

            }


            //START UP PROCESSES
            static Dictionary<string, Int16> LoadWorkingCatalog(Dictionary<string, Int16> dict) //loads all resources from text file to dictionary
            {

                string line;

                StreamReader sr = new StreamReader(@"working-catalog.txt");


                string[] keyAndValue;
                using (sr)
                {
                    while ((line = sr.ReadLine()) != null)
                    {
                        keyAndValue = line.Split(',');
                        dict.Add(keyAndValue[0], Convert.ToInt16(keyAndValue[1])); //add each item to diciontary (working catalog)
                        Array.Clear(keyAndValue, 0, keyAndValue.Length);
                    }




                    return dict;
                }

            }
            static Dictionary<string, Int16> LoadStaticCatalog(Dictionary <string, Int16> dict) //loads all resources from text file to dictionary
            {

            string line;

            StreamReader sr = new StreamReader("static-catalog.txt");

                string[] keyAndValue;
                using (sr)
                {
                    while ((line = sr.ReadLine()) != null)
                    {
                        keyAndValue = line.Split(',');
                        dict.Add(keyAndValue[0], Convert.ToInt16(keyAndValue[1])); //add each item to diciontary (working catalog)
                        Array.Clear(keyAndValue, 0, keyAndValue.Length);
                    }
                    return dict;
                }

            }
            static Dictionary<string, string> LoadStaticIDCatalog(Dictionary<string, string> dict) //loads all resources from text file to dictionary
            {

                string line;

                StreamReader sr = new StreamReader("static-ID-catalog.txt");

                string[] keyAndValue;
                using (sr)
                {
                    while ((line = sr.ReadLine()) != null)
                    {
                        keyAndValue = line.Split(',');
                        dict.Add(keyAndValue[0], keyAndValue[1]); //add each item to diciontary (static ID catalog)
                        Array.Clear(keyAndValue, 0, keyAndValue.Length);
                    }
                    return dict;
                }

            }
            static void LoadResourcesOutList()
            {
                resourcesOutList.Clear();
                string line;
                StreamReader loadResourcesOut = new StreamReader("resourcesOut.txt");
                using (loadResourcesOut)
                {
                    while ((line = loadResourcesOut.ReadLine()) != null)
                    {
                        resourcesOutList.Add(line);
                    }

                }
            }


            //WELCOME AND EXIT
            static void Exit()
            {
                Console.Clear();
                Console.WriteLine("*************** Bootcamp Resources Checkout System ****************\n\n");
                Console.Write("GOOD BYE");
                System.Threading.Thread.Sleep(2000);
                Environment.Exit(0);

            }
            static void Welcome()
            {
                Console.WriteLine("*************** Bootcamp Resources Checkout System ****************\n\n");
                Console.WriteLine("HELLO");
                System.Threading.Thread.Sleep(1200);
            }

            //MAIN METHOD
            static void Main(string[] args)
            {

                //START UP PROCESSES - Loads Saved Data

                //Load data from text files to dicitonaries
                LoadStaticCatalog(staticCatalog);
                LoadStaticIDCatalog(staticIDCatalog);//LoadStatic Catalog method will read text file and assign keys and values to the dictoinary staticCatalog
                LoadWorkingCatalog(workingCatalog); // LoadWorking Catalog method will read text file and assign keys and values to the dictionary workingCatalog
                LoadStudentRoster(studentRoster); //Loads all students to student roster dictionary.. ID = key, first/last name = Value
                LoadResourcesOutList(); //Loads list of resources checked out an by who

                Welcome();
                MenuDisplay();



            }
        }
    }

```