

```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6 using System.IO;
7 using System.Data;
8 using System.Text.RegularExpressions;
9 using System.Windows;
10
11
12 namespace week7_LibraryRevision_1
13 {
14     class Program
15     {
16         //VARIABLES
17
18         //static variables here
19         static string studentID; //holds value of student ID durring processes
20         static string currentName; //holds value of student's name durring processes
21         static string resourcesCheckedOut; //how many resources student currently has checked out
22         static string studentResource1; //holds value of resource student has checked out
23         static string studentResource2; //holds value of resource student has checked out
24         static string studentResource3; //holds value of resource student has checked out
25         static string seperator = ",";
26         static string currentTitleSearch; //holds value of resource during check out process
27         static string currentReturnResourceTitle; //holds value of resource during return process
28         static string[] currentStudentArray = new string[6]; //TODO CAN BE A LIST?
29         static StringBuilder currentStudentHeaderBuilder = new StringBuilder();
30
31         //Declare Dictionaries and Lists
32         static Dictionary<string, Int16> staticCatalog = new Dictionary<string, Int16>(StringComparer.OrdinalIgnoreCase); //Declaring fixed dictionary
33         static Dictionary<string, string> staticIDCatalog = new Dictionary<string, string>(StringComparer.OrdinalIgnoreCase); //Declaring fixed dictionary
34         static Dictionary<string, Int16> workingCatalog = new Dictionary<string, Int16>(StringComparer.OrdinalIgnoreCase); //Declaring mutable catalog dictionary
35         static Dictionary<string, string> studentRoster = new Dictionary<string, string>(StringComparer.OrdinalIgnoreCase); //Declaring fixed catalog dictionary
36         static List<string> resourcesOutList = new List<string>();
37
```

```
38
39     //METHODS
40
41     //MAIN MENU
42     static void MenuDisplay()
43     {
44         //PRINT MENU TO SCREEN
45         Console.Clear();
46         Console.WriteLine("***** Bootcamp Resources      ↗
47         Checkout System *****\n");
48
49         string[] optionsMenu = new string[] { "1 - View Students", ↗
50         "2 - View Available Resources", "3 - Resources Checked Out", ↗
51         "4 - View Student Account", "5 - Check Out Item", "6 - ↗
52         Return Item", "7 - Exit" };
53
54         Console.WriteLine();
55         for (int i = 0; i < optionsMenu.Length; i++)
56         {
57             Console.WriteLine(optionsMenu[i]);
58         }
59         Console.Write("\nChoose a menu item: ");
60
61         string input = Console.ReadLine();
62         input = input.Trim();
63
64         //int menuChoice;
65         int menuChoice;
66         bool res = int.TryParse(input, out menuChoice);
67
68         if (res == false)
69         {
70             Console.Clear();
71             Console.WriteLine("Enter a number from 1 to 7 to make a ↗
72             selection");
73             MenuDisplay();
74         }
75
76         else if (menuChoice < 1 || menuChoice > 7)
77         {
78             Console.Clear();
79             Console.WriteLine("***** Bootcamp Resources      ↗
80             Checkout System *****\n\n");
81
82             Console.WriteLine("Enter a number from 1 to 6 to make a ↗
83             selection");
84             MenuDisplay();
85         }
86     }
```

```
83         switch (menuChoice)
84         {
85             case 1:
86                 ListAllStudentsAlphabetical();
87                 break;
88             case 2:
89                 ListAvailableResourcesAlphabetical2();
90                 break;
91             case 3:
92                 ResroucesOutWithStudentName();
93                 break;
94             case 4:
95             {
96                 VerifyID2();
97                 StudentProfile();
98                 break;
99             }
100             case 5:
101             {
102                 VerifyID2();
103                 StudentCheckOut();
104                 break;
105             }
106             case 6:
107             {
108                 VerifyID2();
109                 ResourceReturn();
110                 break;
111             }
112             case 7:
113                 Exit();
114                 break;
115             default:
116                 break;
117         }
118     }
119 }
120
121
122
123
124 //VERIFY STUDENT ID...validate student ID. Use with check out/return process, view student account process
125 static void VerifyID2() //search for student record. If it exists, sets Student ID Variable. Use for all operations that require student ID
126 {
127
128
129     string choice;
130     string pattern = @"^d{3}$";
131     Regex matchInput = new Regex(pattern, RegexOptions.IgnoreCase);
132     do
```

```
133     {
134         Console.Clear();
135         Console.WriteLine("***** Bootcamp Resources Checkout ↗
136         System *****\n\n");
137         Console.WriteLine("Enter a 3 digit Student ID or M to return to Main ↗
138         Menu: ");
139         choice = Console.ReadLine().ToUpper();
140
141         Match m = matchInput.Match(choice);
142         if (m.Success)
143         {
144             if (!File.Exists(choice + ".txt"))
145             {
146                 Console.WriteLine("That student is not in our records");
147                 Console.Clear();
148                 MenuDisplay();
149             }
150             studentID = choice;
151             break;
152         }
153         else if (choice == "M")
154         {
155             Console.Clear();
156             MenuDisplay();
157         }
158         else
159         {
160
161             choice = "repeat";
162             Console.Clear();
163         }
164     } while (choice == "repeat");
165
166     Console.WriteLine("\nEnter Student ID: ");
167     //string input = Console.ReadLine();
168
169     // check for student in records
170
171
172
173
174     //read each line of a text file and assign to variables
175     StreamReader sr = new StreamReader(studentID + ".txt"); //studentID ↗
176     assigned when user enters
177     using (sr)
178     {
179         string line;
180         int counter = 0;
181
182         //assign values from student text file to currentStudent Array
```

```
182         while (counter < 6)
183         {
184             line = sr.ReadLine();
185             currentStudentArray[counter] = line;
186             counter++;
187         }
188
189         //assign index values from currentStudent Array to individual variables
190         studentID = currentStudentArray[0].ToString();
191         currentName = currentStudentArray[1].ToString();
192         resourcesCheckedOut = currentStudentArray[2].ToString();
193         studentResource1 = currentStudentArray[3].ToString();
194         studentResource2 = currentStudentArray[4].ToString();
195         studentResource3 = currentStudentArray[5].ToString();
196     }
197 }
198
199 //CURRENT STUDENT HEADER
200 static void CurrentStudentHeader()
201 {
202     StringBuilder currentStudentHeaderBuilder = new StringBuilder();
203     Console.WriteLine(currentStudentHeaderBuilder.Append("Current
204         Student: " + currentName));
205     Console.WriteLine();
206 }
207
208 //CHECK OUT PROCESSES
209 static void StudentCheckOut()
210 {
211     Console.Clear();
212     Console.WriteLine("***** Bootcamp Resources Checkout
213         System *****\n\n");
214     CurrentStudentHeader();
215
216     //check resourcesCheckedOut variable...if >2, student may not check
217     out books, return to menu
218     if (int.Parse(resourcesCheckedOut) > 2)
219     {
220
221         StringBuilder maxResourcesOut = new StringBuilder();
222         maxResourcesOut.Append(currentName).Append(" has checked out the
223             maximum number of resources.");
224         Console.WriteLine(maxResourcesOut);
225
226         Console.Write("\nPress any key to return to Main Menu");
227         Console.ReadKey();
228         MenuDisplay();
229     }
```

```
229
230         TitleSearchAvailabilityAlt(); //search for title availability
231
232         MenuDisplay();
233
234
235     }
236     static void TitleSearchAvailabilityAlt()
237     {
238
239         //Console.WriteLine("Enter resource ID of item to checkout or V to  ↗
240             view resource IDs");
241         //string userInput = Console.ReadLine();
242
243         string choice = null;
244
245         do
246         {
247             Console.Clear();
248             Console.WriteLine("***** Bootcamp Resources Checkout  ↗
249                 System *****\n\n");
250             CurrentStudentHeader();
251
252             Console.Write("Enter resource ID of item to checkout or V to  ↗
253                 view resource IDs: ");
254             choice = Console.ReadLine().ToUpper().Trim();
255             if (choice == "V")
256             {
257                 Console.Clear();
258                 Console.WriteLine("***** Bootcamp Resources  ↗
259                     Checkout System *****\n\n");
260                 CurrentStudentHeader();
261
262                 ListResourceWithID();
263                 Console.Write("\nEnter resource ID of item to checkout or M  ↗
264                     for Main Menu: ");
265                 choice = Console.ReadLine().ToUpper();
266                 if (choice == "M")
267                 {
268                     Console.Clear();
269                     MenuDisplay();
270                 }
271                 else if (staticIDCatalog.ContainsKey(choice))
272                 {
273                     break;
274                 }
275                 else
276                 {
277                     choice = "repeat";
278                     Console.Clear();
279                 }
280             }
281         }
282     }
```

```
276     }
277     else if (staticIDCatalog.ContainsKey(choice))
278     {
279
280         break;
281     }
282
283     else
284     {
285
286         choice = "repeat";
287         Console.Clear();
288     }
289 } while (choice == "repeat");
290
291 currentTitleSearch = staticIDCatalog[choice];
292
293 //check to see if there is a copy available to check out.
294 if (workingCatalog[currentTitleSearch] == 0)
295 {
296     Console.Clear();
297     Console.WriteLine("There are no copies of " + currentTitleSearch +
298         " availalbe at this time\n");
299     //turn this do loop into a method?
300     //string choice;
301     do
302     {
303         Console.WriteLine("Enter S to search again, or M to return
304             to the Main Menu\n: ");
305         choice = Console.ReadLine().ToUpper();
306         if (choice == "M")
307         {
308             Console.Clear();
309             MenuDisplay();
310         }
311         else if (choice == "S")
312         {
313             TitleSearchAvailabilityAlt();
314         }
315     }
316     else
317     {
318         choice = "repeat";
319         Console.Clear();
320     }
321 } while (choice == "repeat");
322
323 // TODO CONFIRM STUDENT WANTS TO CHECKOUT HERE?
324
325 //decrement resource availability in workingCatalog dictionary
```

```
326         if (workingCatalog.ContainsKey(currentTitleSearch)) //WTF
327         {
328             workingCatalog[currentTitleSearch] -= 1;
329
330             //save working Catalog to File
331             SaveWorkingCatalogToFile();
332
333             //save resource to student file and write to ResourcesOutList
334             SaveResourceToStudentFile();
335             SaveToResourcesOutList();
336             Console.Clear();
337             Console.WriteLine("***** Bootcamp Resources Checkout  ↗
338                 System *****\n\n");
339             CurrentStudentHeader();
340
341             Console.WriteLine(currentName + " has checked out " +      ↗
342                 currentTitleSearch + "\n"); // TODO I would like this title to ↗
343                 come from the Array for correct formatting
344
345             if (int.Parse(resourcesCheckedOut) > 2)
346             {
347                 Console.WriteLine("Press any key to return to Main Menu");
348                 Console.ReadKey();
349                 Console.Clear();
350                 MenuDisplay();
351             }
352
353             //offer option to check out again..turn into method?
354             //string choice;
355             do
356             {
357                 Console.WriteLine("\nEnter S check out another item, or M to ↗
358                     return to the Main Menu: \n");
359                 choice = Console.ReadLine().ToUpper();
360                 if (choice == "M")
361                 {
362                     Console.Clear();
363                     MenuDisplay();
364                 }
365                 else if (choice == "S")
366                 {
367                     TitleSearchAvailabilityAlt();
368                 }
369                 else
370                 {
371                     choice = "repeat";
372                     Console.Clear();
373                 }
374             } while (choice == "repeat");
```



```
374     }
375     }
376 }
377 static void SaveResourceToStudentFile()
378 {
379     //find resource in currentStudent array
380     for (int i = 3; i < currentStudentArray.Length; i++)
381     {
382         if ((currentStudentArray[i] != "-")) //extra parenthesis? TODO
383         {
384             continue;
385         }
386
387         if ((currentStudentArray[i] == "-")) //extra parenthesis? TODO
388         {
389             //assign currentTitleSearch value to array
390             //currentStudent[i] = currentTitleSearch;
391             //assign currentTitleSearch value to currentStudent list
392             if (i == 3)
393             {
394                 currentStudentArray[i] = currentTitleSearch;
395                 studentResource1 = currentTitleSearch;
396             }
397             else if (i == 4)
398             {
399                 currentStudentArray[i] = currentTitleSearch;
400                 studentResource2 = currentTitleSearch;
401             }
402             else if (i == 5)
403             {
404                 currentStudentArray[i] = currentTitleSearch;
405                 studentResource3 = currentTitleSearch;
406             }
407
408             //math to increase number of books checked out
409             int x = int.Parse(resourcesCheckedOut);
410             x++;
411             string y = x.ToString();
412             currentStudentArray[2] = y;
413
414             //update number of resources student has checked out
415             resourcesCheckedOut = y;
416
417             break;
418         }
419     }
420 }
421 //write updated student information to file
```

```
426         using (StreamWriter SaveStudentFile = new StreamWriter ↗
427             (studentID + ".txt")) //delete student text file
428         {
429         }
430
431         using (StreamWriter sw = File.AppendText(studentID + ↗
432             ".txt")) //write new values to student text tile
433         {
434             sw.WriteLine(studentID);
435             sw.WriteLine(currentName);
436             sw.WriteLine(resourcesCheckedOut);
437             sw.WriteLine(studentResource1);
438             sw.WriteLine(studentResource2);
439             sw.WriteLine(studentResource3);
440         }
441     }
442
443     //WRITE RESOURCE TO "RESOURCESOUT" LIST
444     static void SaveToResourcesOutList()
445     {
446
447         //use stringBuilder to concat studentName and currentTitleSearch
448         StringBuilder resourceAndStudentCSV = new StringBuilder();
449         resourceAndStudentCSV.Append(currentTitleSearch).Append ↗
450             (seperator).Append(currentName);
451         string resourceAndStudent = resourceAndStudentCSV.ToString();
452
453         //add currently checked out resource to resourcesOutList
454         resourcesOutList.Add(resourceAndStudent);
455         StreamWriter saveResourcesOutToText = new StreamWriter ↗
456             ("resourcesOut.txt", true);
457         using (saveResourcesOutToText)
458         {
459             saveResourcesOutToText.WriteLine(resourceAndStudent);
460         }
461     }
462
463     //SAVE workingCatalog TO FILE (after checkout or return..updates ↗
464         resources checked out/available)
465     static void SaveWorkingCatalogToFile()
466     {
467         using (StreamWriter SaveWorkingCatatlog = new StreamWriter("working- ↗
468             catalog.txt"))
469         {
470             foreach (KeyValuePair<string, Int16> kvp in workingCatalog)
471             {
472                 StringBuilder workingCatalogBuildString = new StringBuilder ↗
473                     ();
```

```

471         string saveWorkingCatalog =
472             (workingCatalogBuildString.Append(kvp.Key).Append
473             (seperator).Append(kvp.Value)).ToString();
474             SaveWorkingCatalog.WriteLine(saveWorkingCatalog);
475         }
476     }
477     //RETURN PROCESS
478     static void ResourceReturn() //takes student ID as argument
479     {
480
481         //does student have any resources out?
482
483         if ((int.Parse(resourcesCheckedOut) < 1 ))
484         {
485             Console.Clear();
486             Console.WriteLine("***** Bootcamp Resources Checkout
487             System *****\n\n");
488             CurrentStudentHeader();
489             Console.WriteLine(currentName + " has 0 resrouces checked
490             out.");
491             Console.Write("\nPress any key to return to Main Menu");
492             Console.ReadKey();
493             Console.Clear();
494             MenuDisplay();
495         }
496
497         CurrentStudentResourcesCheckedOut();
498         ReturnResourceToWorkingCatalog();
499         ReturnResourceToStudentFile();
500         RemoveResourceFromResourceOutList();
501         Console.Clear();
502         Console.WriteLine("***** Bootcamp Resources Checkout
503         System *****\n\n");
504         CurrentStudentHeader();
505
506         StringBuilder hasReturned = new StringBuilder();
507         hasReturned.Append(currentName).Append(" has returned ").Append
508         (currentReturnResourceTitle);
509         Console.WriteLine(hasReturned);
510
511         string choice;
512         do
513         {
514             Console.WriteLine("\nEnter R to return another item for " +
515             currentName + " or M to return to the Main Menu\n");
516             choice = Console.ReadLine().ToUpper();
517             if (choice == "R")
518             {
519                 Console.Clear();

```

```
516         ResourceReturn();
517     }
518     else if (choice == "M")
519     {
520         MenuDisplay();
521     }
522
523     else
524     {
525
526         choice = "repeat";
527         Console.Clear();
528     }
529 } while (choice == "repeat");
530
531
532 MenuDisplay();
533 ;
534
535 } //what is this?
536 static void CurrentStudentResourcesCheckedOut()
537 {
538     Console.Clear();
539
540     Dictionary<string, string> returnOptions = new Dictionary<string,  ↗
541         string>();
542     int counter = 1;
543     for (int i = 3; i < currentStudentArray.Length; i++)
544     {
545         if ((currentStudentArray[i] == "-")) //does this have an extra  ↗
546             parenthesis? TODO
547         {
548             continue;
549         }
550         if ((currentStudentArray[i] != "-")) //extra parenthesis? TODO
551         {
552             returnOptions.Add(counter.ToString(),  ↗
553                 currentStudentArray[i]);
554             counter++;
555         }
556     }
557
558     //choose which item to return
559
560     string input;
561     do
562     {
563         Console.Clear();
564         Console.WriteLine("***** Bootcamp Resources Checkout  ↗
565             System *****\n\n");
566         CurrentStudentHeader();
```

```

564         Console.WriteLine("Resources checked out:\n");
565         foreach (KeyValuePair<string,string> kvp in returnOptions)
566         {
567             Console.WriteLine(kvp.Key + ". " + kvp.Value);
568         }
569
570
571         Console.Write("\n\nEnter the number of the item you would like  ➤
to return: ");
572         input = Console.ReadLine().Trim();
573         if (returnOptions.ContainsKey(input))
574         {
575             string value;
576             if (returnOptions.TryGetValue(input, out value))
577             {
578                 currentReturnResourceTitle = value;
579                 Console.WriteLine("you chose to return: " +  ➤
currentReturnResourceTitle);    //DEBUG
580             }
581         }
582         } while (!(returnOptions.ContainsKey(input)));
583
584     }
585     static void ReturnResourceToWorkingCatalog() //saves returned resource  ➤
to txt file
586     {
587         Console.Clear();
588         Console.WriteLine("***** Bootcamp Resources Checkout  ➤
System *****\n\n");
589         CurrentStudentHeader();
590
591         //This increments available parameter in workingCatalog dictionary
592         if (workingCatalog.ContainsKey(currentReturnResourceTitle)) //WTF
593         {
594             workingCatalog[currentReturnResourceTitle] ++;
595             Console.WriteLine(currentName + " has returned " +  ➤
currentReturnResourceTitle + ".");
596
597             //save working Catalog to File
598             SaveWorkingCatalogToFile();
599         }
600     }
601     static void ReturnResourceToStudentFile() //saves returned resource to  ➤
student txt file, updates currentStudent Array
602     {
603
604         //find resource in currentStudent array
605         for (int i = 3; i < currentStudentArray.Length; i++)
606         {
607             if ((currentStudentArray[i] != currentReturnResourceTitle)) //  ➤
extra parenthesis? TODO
608         {

```

```
609         continue;
610     }
611
612
613     if ((currentStudentArray[i] == currentReturnResourceTitle)) // ↗
        extra parenthesis? TODO
614     {
615         //reset array resource value to "-"
616         //reset student resource variable to "-"
617         if (i == 3)
618         {
619             currentStudentArray[i] = "-";
620             studentResource1 = "-";
621
622         }
623         else if (i == 4)
624         {
625
626             currentStudentArray[i] = "-";
627             studentResource2 = "-";
628         }
629         else if (i == 5)
630         {
631
632             currentStudentArray[i] = "-";
633             studentResource3 = "-";
634         }
635
636         //math to decrease number of books checked out
637         int x = int.Parse(resourcesCheckedOut);
638         x--;
639         string y = x.ToString();
640         currentStudentArray[2] = y;
641
642         //update number of resources student has checked out
643         resourcesCheckedOut = y;
644
645         break;
646     }
647
648 }
649 //write updated student information to file
650 using (StreamWriter SaveStudentFile = new StreamWriter(studentID + ↗
        ".txt")) //delete student text file
651 {
652
653 }
654
655 using (StreamWriter sw = File.AppendText(studentID + ".txt")) // ↗
        write new values to student text file
656 {
657     sw.WriteLine(studentID);
```

```
658         sw.WriteLine(currentName);
659         sw.WriteLine(resourcesCheckedOut);
660         sw.WriteLine(studentResource1);
661         sw.WriteLine(studentResource2);
662         sw.WriteLine(studentResource3);
663     }
664 }
665 }
666
667 //REMOVE RESOURCE FROM "RESOURCESOUT" LIST
668 static void RemoveResourceFromResourceOutList()
669 {
670     //find index of resource and student name
671
672     string returnResourceAndStudent = currentReturnResourceTitle +
        separator + currentName;
673
674     for (int i = 0; i < resourcesOutList.Count; i++)
675     {
676         if (resourcesOutList[i].ToString().Equals
            (returnResourceAndStudent, StringComparison.CurrentCultureIgnore
            eCase))
677         {
678             resourcesOutList.RemoveAt(i);
679             File.Delete("resourcesOut.txt");
680             StreamWriter updateResourcesOutTextFile = new StreamWriter
            ("resourcesOut.txt"); //TODO FIX!! not writing to file
            properly
681             using (updateResourcesOutTextFile)
682             {
683                 foreach (string item in resourcesOutList)
684                 {
685                     updateResourcesOutTextFile.WriteLine(item);
686                 }
687             }
688         }
689     }
690
691 }
692 }
693
694 }
695
696 //LIST STUDENTS
697 static void StudentProfile()
698 {
699     Console.Clear();
700     Console.WriteLine("***** Bootcamp Resources Checkout
        System *****\n\n");
701     Console.WriteLine("\nStudent ID: " + studentID);
702     Console.WriteLine("Name: " + currentName);
703     Console.WriteLine("\n\n" + resourcesCheckedOut + " resources checked
```

```

        out: \n");
704         int counter = 1;
705         for (int i = 3; i < currentStudentArray.Length; i++)
706         {
707             if ((currentStudentArray[i] != "-")) //extra parenthesis? TODO
708             {
709                 Console.WriteLine(counter.ToString() + ". " +
                                   currentStudentArray[i]);
710                 counter++;
711             }
712         }
713     }
714     Console.Write("\nPress any key to return to Main Menu");
715     Console.ReadKey();
716     MenuDisplay();
717 }
718
719 static DataTable CreateStudentRosterTable(Dictionary<string, string>
720 dict) //creates table from student roster dictionary, returns a table
721 {
722     DataTable table = new DataTable();
723     table.Columns.Add("Student ID", typeof(string)); //converting a
724     //dictionary to a table will always have only two columns..what if I
725     //want to combine dictionaries? should I just store this all in a
726     //table?
727     table.Columns.Add("Student Name", typeof(string));
728
729     foreach (KeyValuePair<string, string> kvp in dict) //adds key and
730     //value of dictionary to table
731     {
732         table.Rows.Add(kvp.Key, kvp.Value);
733     }
734     //after the for each loop, a table exists with all student Id and
735     //Name in rows
736     return table;
737 }
738
739 static void ListAllStudentsAlphabetical()
740 {
741     Console.Clear();
742     Console.WriteLine("***** Bootcamp Resources Checkout
743     System *****\n\n");
744
745     //DICTIONARY TO DATATABLE
746     DataTable table = CreateStudentRosterTable(studentRoster); //returns
747     //a table of students and id to new table
748
749     //create dataview object of table so I can sort it
750     DataView view = new DataView(table);
751
752     //sorts dataview object by column named Name - ascending
753     view.Sort = "Student Name ASC";
754 }
755

```



```

746         //print columnn headers
747         foreach (DataColumn column in table.Columns)
748         {
749             Console.Write(column.ColumnName + "\t");
750         }
751         Console.WriteLine();
752         Console.WriteLine();
753
754         //print sorted data table row by row
755         foreach (DataRowView row in view)
756         {
757             Console.WriteLine(" {0}\t\t{1}", row[0], row[1]);
758         }
759         Console.WriteLine("\n");
760         Console.Write("Press any key to return to Main Menu");
761         Console.ReadKey();
762         Console.Clear();
763         MenuDisplay();
764         //return dict;
765     }
766     static Dictionary<string, string> LoadStudentRoster(Dictionary<string,
767     string> dict) //loads all student and id from text file to dictionary
768     {
769         string line;
770         string[] keyAndValue;
771         //List<string> students = new List<string>();
772         StreamReader sr = new StreamReader(@"student-roster.txt");
773         using (sr)
774         {
775             while ((line = sr.ReadLine()) != null)
776             {
777                 keyAndValue = line.Split(',');
778                 dict.Add(keyAndValue[0], keyAndValue[1]);
779                 Array.Clear(keyAndValue, 0, keyAndValue.Length);
780             }
781             return dict;
782         }
783     }
784 }
785
786
787 //LIST AVAIALABLE RESOURCES
788 static DataTable CreateAvailableResourceTable(Dictionary<string, Int16>
789 dict) //creates table from workingCatalog dictionary, returns a table
790 {
791     DataTable table = new DataTable();
792     table.Columns.Add("Available", typeof(Int16));
793     table.Columns.Add("Resource", typeof(string)); //converting a
794     dictionary to a table will always have only two columns..what if I
795     want to combine dictionaries? should I just store this all in a
796     table?

```

```
793
794     foreach (KeyValuePair<string, Int16> kvp in dict) //adds key and value of dictionary to table
795     {
796         table.Rows.Add(kvp.Value, kvp.Key);
797     }
798     //after the for each loop, a table exists with all student Id and Name in rows
799     return table;
800 }
801 static void ListAvailableResourcesAlphabetical2()
802 {
803     Console.Clear();
804     Console.WriteLine("***** Bootcamp Resources Checkout System *****\n\n");
805
806     //DICTIONARY TO DATATABLE
807     DataTable table = CreateAvailableResourceTable(workingCatalog); // returns a table of students and id to new table
808
809     //create dataview object of table so I can sort it
810     DataView view = new DataView(table);
811
812     //sorts dataview object by column named Name - ascending
813     view.Sort = "Resource ASC";
814
815     //print column headers
816
817
818     foreach (DataColumn column in table.Columns)
819     {
820         Console.Write(column.ColumnName + "\t\t");
821     }
822     Console.WriteLine();
823     Console.WriteLine();
824
825     //print sorted data table row by row
826     foreach (DataRowView row in view)
827     {
828         Console.WriteLine(" {1}\t" + "\t{0}", row[1], row[0]);
829     }
830
831     Console.WriteLine("\n");
832     Console.Write("Press any key to return to Main Menu");
833     Console.ReadKey();
834     Console.Clear();
835     MenuDisplay();
836     //return dict;
837 }
838
839 //LIST RESOURCES OUT + STUDENT NAME
840 static void ResroucesOutWithStudentName()
```

```
841     {
842         Console.Clear();
843         Console.WriteLine("Resources Checked Out:\n");
844         resourcesOutList.Sort();
845         for (int i = 0; i < resourcesOutList.Count; i++)
846         {
847             Console.WriteLine(resourcesOutList[i]);
848         }
849
850         Console.Write("\n\nPress any key to return to Main Menu");
851         Console.ReadKey();
852         Console.Clear();
853         MenuDisplay();
854     }
855
856     //LIST RESOURCE IDs
857     static DataTable CreateResourceIDTable(Dictionary<string, string> dict) //creates table from static-ID-catalog dictionary
858     {
859         DataTable table = new DataTable();
860         table.Columns.Add("Resource ID", typeof(string));
861         table.Columns.Add("Resource", typeof(string)); //converting a
862             dictionary to a table will always have only two columns..what if I
863             want to combine dictionaries? should I just store this all in a
864             table?
865
866         foreach (KeyValuePair<string, string> kvp in dict) //adds key and
867             value of dictionary to table
868         {
869             table.Rows.Add(kvp.Key, kvp.Value);
870         }
871         //after the for each loop, a table exists with all student Id and
872             Name in rows
873         return table;
874     }
875     static void ListResourceWithID() //sorts alphabetical and displays
876     {
877         Console.Clear();
878         Console.WriteLine("***** Bootcamp Resources Checkout
879             System *****\n\n");
880
881         //DICTIONARY TO DATATABLE
882         DataTable table = CreateResourceIDTable(staticIDCatalog); //returns
883             a table of students and id to new table
884
885         //create dataview object of table so I can sort it
886         DataView view = new DataView(table);
887
888         //sorts dataview object by column named Name - ascending
889         view.Sort = "Resource ASC";
890
891         //print column headers
```

```
885
886
887     foreach (DataColumn column in table.Columns)
888     {
889         Console.Write(column.ColumnName + "\t\t");
890     }
891     Console.WriteLine();
892     Console.WriteLine();
893
894     //print sorted data table row by row
895     foreach (DataRowView row in view)
896     {
897         Console.WriteLine(" {1}\t" + "\t{0}", row[1], row[0]);
898     }
899
900 }
901
902
903
904 //START UP PROCESSES
905 static Dictionary<string, Int16> LoadWorkingCatalog(Dictionary<string,  ↗
906     Int16> dict) //loads all resources from text file to dictionary
907 {
908     string line;
909
910     StreamReader sr = new StreamReader(@"working-catalog.txt");
911
912
913     string[] keyAndValue;
914     using (sr)
915     {
916         while ((line = sr.ReadLine()) != null)
917         {
918             keyAndValue = line.Split(',');
919             dict.Add(keyAndValue[0], Convert.ToInt16(keyAndValue  ↗
920                 [1])); //add each item to dictionary (working catalog)
921             Array.Clear(keyAndValue, 0, keyAndValue.Length);
922         }
923
924
925
926         return dict;
927     }
928
929 }
930 static Dictionary<string, Int16> LoadStaticCatalog(Dictionary <string,  ↗
931     Int16> dict) //loads all resources from text file to dictionary
932 {
933     string line;
```

```
934
935     StreamReader sr = new StreamReader("static-catalog.txt");
936
937     string[] keyAndValue;
938     using (sr)
939     {
940         while ((line = sr.ReadLine()) != null)
941         {
942             keyAndValue = line.Split(',');
943             dict.Add(keyAndValue[0], Convert.ToInt16(keyAndValue[1])); //add each item to dictionary (working catalog)
944             Array.Clear(keyAndValue, 0, keyAndValue.Length);
945         }
946         return dict;
947     }
948
949 }
950 static Dictionary<string, string> LoadStaticIDCatalog(Dictionary<string, string> dict) //loads all resources from text file to dictionary
951 {
952
953     string line;
954
955     StreamReader sr = new StreamReader("static-ID-catalog.txt");
956
957     string[] keyAndValue;
958     using (sr)
959     {
960         while ((line = sr.ReadLine()) != null)
961         {
962             keyAndValue = line.Split(',');
963             dict.Add(keyAndValue[0], keyAndValue[1]); //add each item to dictionary (static ID catalog)
964             Array.Clear(keyAndValue, 0, keyAndValue.Length);
965         }
966         return dict;
967     }
968
969 }
970 static void LoadResourcesOutList()
971 {
972     resourcesOutList.Clear();
973     string line;
974     StreamReader loadResourcesOut = new StreamReader("resourcesOut.txt");
975     using (loadResourcesOut)
976     {
977         while ((line = loadResourcesOut.ReadLine()) != null)
978         {
979             resourcesOutList.Add(line);
980         }
981     }
```

```
982     }
983 }
984
985
986 //WELCOME AND EXIT
987 static void Exit()
988 {
989     Console.Clear();
990     Console.WriteLine("***** Bootcamp Resources Checkout
991         System *****\n\n");
992     Console.Write("GOOD BYE");
993     System.Threading.Thread.Sleep(2000);
994     Environment.Exit(0);
995 }
996 static void Welcome()
997 {
998     Console.WriteLine("***** Bootcamp Resources Checkout
999         System *****\n\n");
1000     Console.WriteLine("HELLO");
1001     System.Threading.Thread.Sleep(1200);
1002 }
1003 //MAIN METHOD
1004 static void Main(string[] args)
1005 {
1006
1007     //START UP PROCESSES - Loads Saved Data
1008
1009     //Load data from text files to dicitonaries
1010     LoadStaticCatalog(staticCatalog);
1011     LoadStaticIDCatalog(staticIDCatalog); //LoadStatic Catalog method
1012         will read text file and assign keys and values to the dictoinary
1013         staticCatalog
1014     LoadWorkingCatalog(workingCatalog); // LoadWorking Catalog method
1015         will read text file and assign keys and values to the dictionary
1016         workingCatalog
1017     LoadStudentRoster(studentRoster); //Loads all students to student
1018         roster dictionary.. ID = key, first/last name = Value
1019     LoadResourcesOutList(); //Loads list of resources checked out an by
1020         who
1021
1022     Welcome();
1023     MenuDisplay();
1024 }
```