**Project Report: 11/27/23**

**(Group 11)**

Our end of the year project for the 2023 Fall academic year for Programming Two, titled, *“Eagle’s Library Management System”*. Each respective contributor for this project includes:

**Mikayla Ries & Nash Morrison**

**COP 3003 - CRN: 83937**

**FGCU - Dr. Ciris**

1. **Project Goal**

The Eagle’s Library Management System was a complete overhaul of an already created project found on [*the following website.*](http://www.cppforschool.com/project/library-management-system.html)Mikayla was the first to find this source of inspiration, which highlighted OOP (Object-Oriented-Programming) concepts that were throughout the code which coincided with the material covered in classroom lessons. Both of us decided that this would be a great project to represent our skills learned throughout the semester, however we found out early on that our source of inspiration was in fact “broken.” The whole concept behind this project is to store classified private data within files and those files, retrieved without the worry of a potential data leak. What we realized was that certain functions were not reading and writing to files, with user input not storing our private character data members which meant we had to begin from nothing and build upon the given source code. This alone established this as a completely new project that we had to reverse engineer to figure out how exactly everything should function. Having only touched on file streaming briefly at the end of the semester, both Mikayla and I had to learn everything from the ground up to get our program working flawlessly. The Eagle’s Library Management System, designed to complete the following:

* Allow the user to enter the record of new books by book number, name of the book, and the author’s name.
* Allow the user to modify and delete book records.
* Retrieve the details of books available within the library.
* Create student user accounts with a name and an ID number.
* Modify student user account names.
* Allow users to be issued a book and keep record of the book issued on file.
* Able to track the return of the books and access a late fee after a 15-day window.
* Display a/all books, a/all students, and a specific student account, held on file and within confidential data members found within classes.
* The program executes all within the console.

1. **Methods/Approach**

Our initial inspiration had all the code found within one .CPP file. Since the topics covered in our course heavily cover OOP concepts, we immediately had to split up the project into three separate parts. We were able to seamlessly utilize both encapsulation and inheritance within our classes and associated functions with the incorporation of a .H file alongside an implementation.CPP file. Getter and setter functions, used in both of our Library Book and Student classes allow us to retrieve the data stored within our confidential data members so that only the software engineer has access to that information.

With our project driven through the console, we had to design menus within menus without causing any crashes which means we had to create conditions that were able to repeat itself. We achieved this in unique ways. Switch statements to control the flow of both our student and book menus, while a for-loop is controlling the main menu that executes at the start of the program. Do/while loops are frequent when we are writing to file and are awaiting the user to complete their inputs.

A diagram of a flowchart

Description automatically generated

Figure : Detailed Flowchart

“*Figure A”* shown above, outlines our algorithm and procedure in detail. A big component of our algorithm deals with pointers, which was a topic covered earlier in the semester. Instead of storing the data within actual arrays or vectors and then indexing those respectively, our program is dealing with specific pointer memory allocation within each private character data member. Say for example we want to modify a student on file. Our algorithm allows us to specifically go to that location in memory that we want to modify within our data members, modify from user input and only touching that data found within that block of memory. This provided us with a deeper understanding of why pointers are so important. Both of us stumbled upon a concept shown later in our curriculum, so it was incredible to get our feet wet on the material.

The program functions by user input. When a user wants to create/modify a book or a student, we take the proper precautions to ensure that the user would not mess up. By using cin.getline() with our classified data members, we were able to make sure that we got every bit of data the user input without any missing characters.

A screenshot of a computer program

Description automatically generated

Figure : Example of cin.getline()

File streaming is also another large component of our project. Both our Library Book and Student class must be able to read and write to files to make this program successful. We initially tried implementing .txt files within our program but found that it was causing different errors, so we settled on .Dat files to manage the workload. Without pointer memory allocation, we would not have been able to find our specific information stored in our data members to read or write to files. The flipside to this is that we also needed to be able to find specific locations within .Dat files to add or modify the data, so we are keeping track and dealing with two separate character pointer locations simultaneously. For example:

A computer screen shot of a program

Description automatically generated

Figure : Pointer Memory Location Example

Say the user wants to look at the specific details of a book. The user inputs the book number within the book menu. Our while loop is using a character pointer to find both the book number in our classified data member as well as checking to see if the book is on file. Our strcmp() function returns Boolean values which allows us to double check to make sure that the book is in both locations in memory. This algorithm is critical and would otherwise not allow the program to run properly with this unique setup of code.

1. **Results**

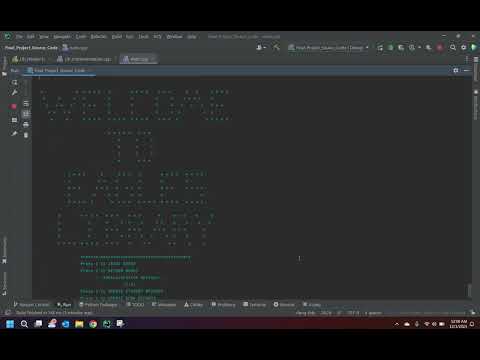
**[](https://www.youtube.com/embed/Aw79hdzMPPE?feature=oembed)**

Figure : Live Demonstration of Project

*“Figure D”* shown above is a recorded demonstration of our project that highlights every single concept, input, and output. Key concepts include keeping the user within each respective menu. Providing exception handling at every turn of the project while providing a nice, formatted intro message every time the user returns to the main menu.

1. **Discussion & Conclusion**

In conclusion, there were difficulties and limitations that we encountered throughout the course of our project. With both of us having different operating systems (Windows vs. Mac), functions did not translate well between systems when we were working on code separately. Our intro() function would not work due to a “system” line of code that changed the color in the console to blue as well as the getch() function on a Mac which created infinite loops and missing character values. Another issue that we ran into had to deal with our use of strcmp() and strcmpi() functions. Initially we tried to establish our personal data members as string variables. These two functions only allow us to pass constant character pointer values which would not allow us to compare either our book number to the number written on file or the student id number to the number on file. So, we had to go back, redefine our functions to character values so that our program would run smoothly. This was our main issue when it came to having issues with saving our data to .txt files. I have run the code with .txt and it runs just as perfectly which has me to believe that switching to character values made all the difference. We also tried experimenting with derived classes, specifically throwing all our functions not found within our Library Book and student classes into another class from which we could just derive. This caused more problems than we initially anticipated. Data would disappear when dealing with file streaming/deriving this class and it just made more sense to keep them on their own and just use inheritance and encapsulation with our other two classes.

Limitations of the project include not using arrays or vectors to store our classified data members. We decided to take the tough rough and deal with pointer memory. We could have implemented a GUI, but I feel as though we would not have enough time with how difficult this project got at times. Our program also only allows a user to have one book issued to them at any given time. This is not how a proper library works, which always gives us the opportunity to go back and revise our code.

The take home message is the importance of pointer memory location and how critical of a tool it can be for a software engineer. Having the ability to have one variable, large enough to hold whatever you like and then specifically go to that location in memory, change what you would like and go about your day is just profound to me. This was one of the most interesting topics I have covered in my education so far, because of how in depth the concept can become, and I would like to see this incorporated earlier on in FGCU’s Software Engineering Program.

1. **Workload Distribution**

We distributed the workload equally down the middle between the two of us. I managed the Library Book class and all its components while Mikayla oversaw the student class and its respective functions. Mikayla took on the responsibility of the PowerPoint while I drafted the report. Communication between both of us was solid throughout the entire project. Even if we were stuck on our own assignments, we would still help each other for the sake of the completion of the project. I can confidently say that we worked well as a collective, and that this project would not have been as easy if the two of us were not a group.

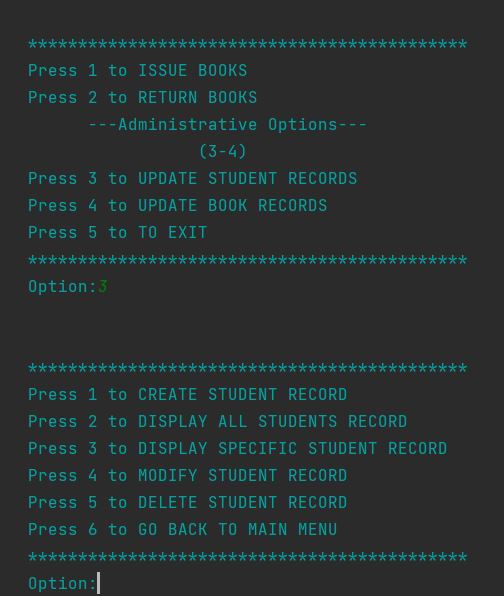
1. **References**
2. <https://www.geeksforgeeks.org/>
3. **Appendix**
4. [Inspiration Source Code](http://www.cppforschool.com/project/library-management-system.html)
5. [GitHub Project Code Link](https://github.com/NWMorrison/Eagles_Library_Management_System/tree/main/Final_Project_Source_Code) (Our Code)
6. [Flow Chart](https://github.com/NWMorrison/Eagles_Library_Management_System/blob/main/Flowchart.drawing.png)
7. [Live Demonstration Youtube Link](https://youtu.be/Aw79hdzMPPE)

**Demonstration**

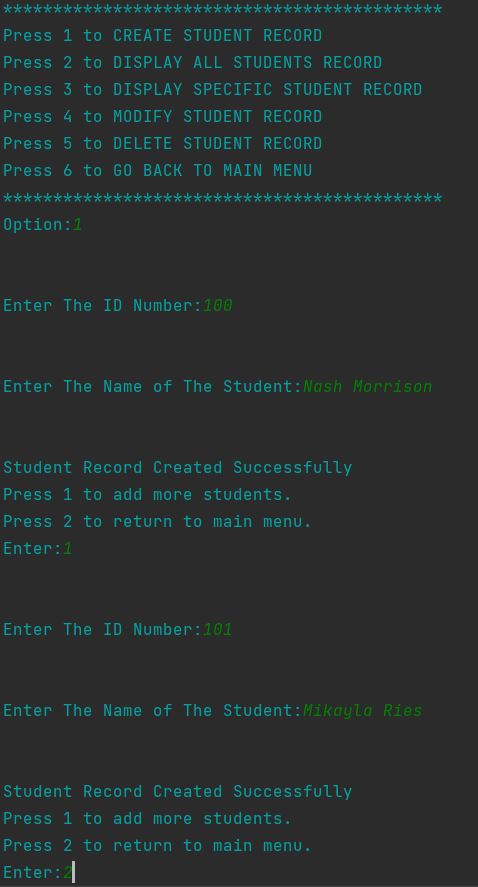
1. **Main Menu**

****

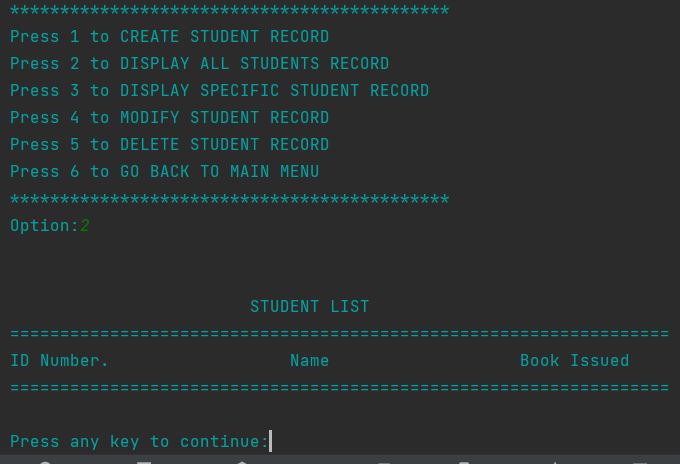
1. **Student Menu**

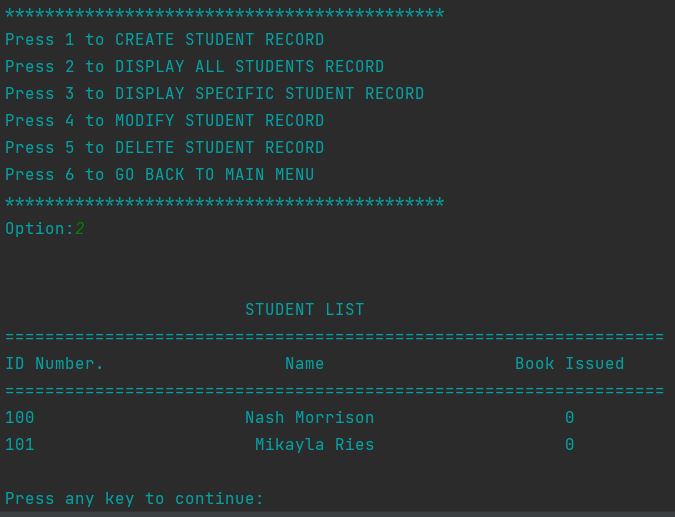
****

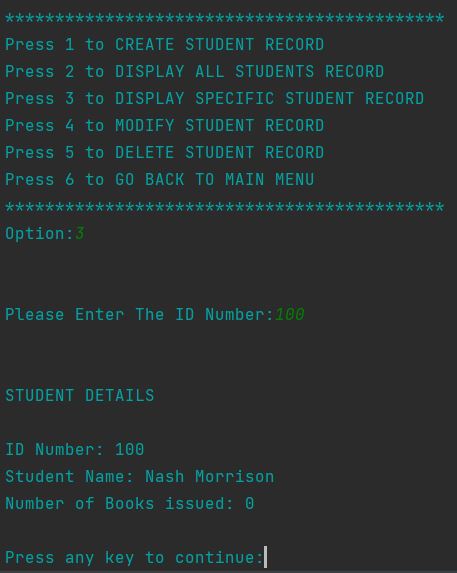
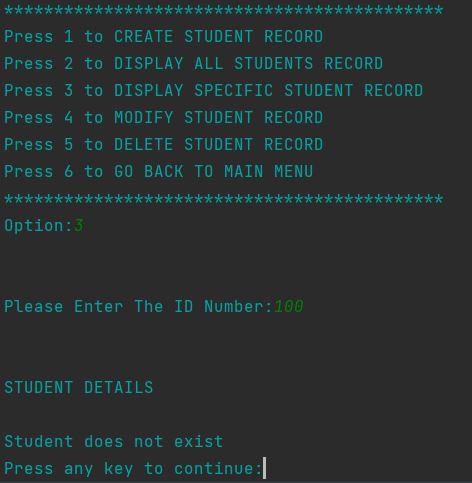
1. **Create Student Record**

****

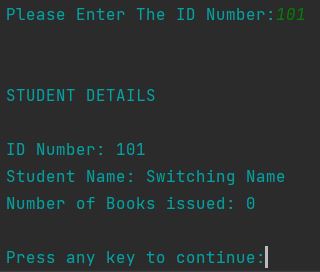
1. **Display All Student Records**

****

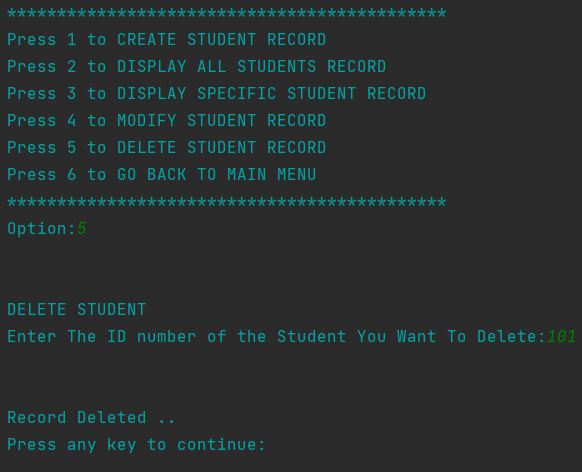
****

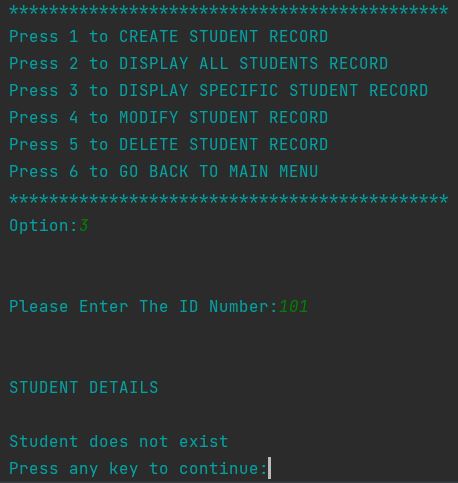
1. **Display Specific Student Record**
2. **Modify Student Record**

****

****

1. **Delete Student Record**

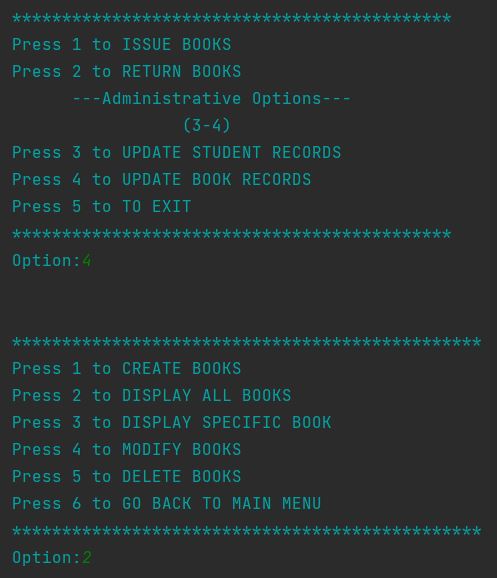
****

****

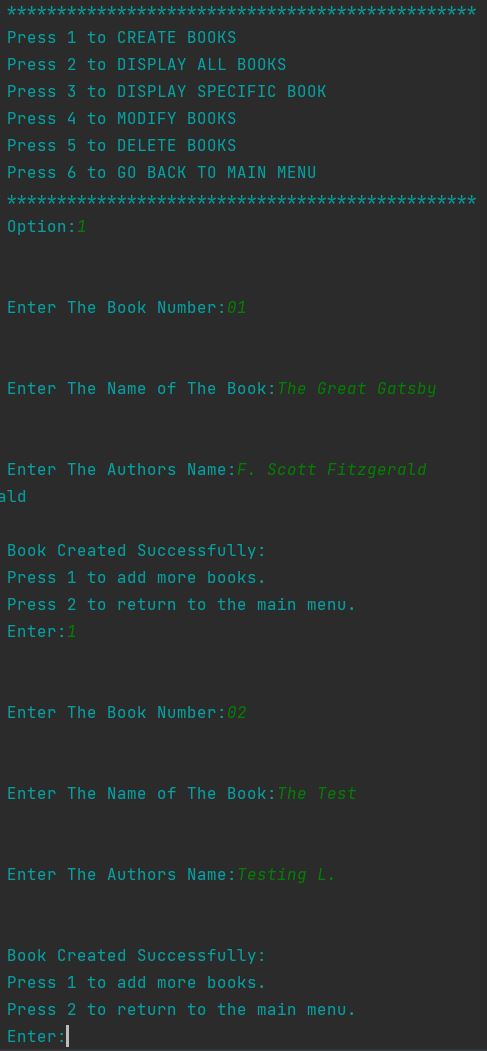
1. **Return To Main Menu**

****

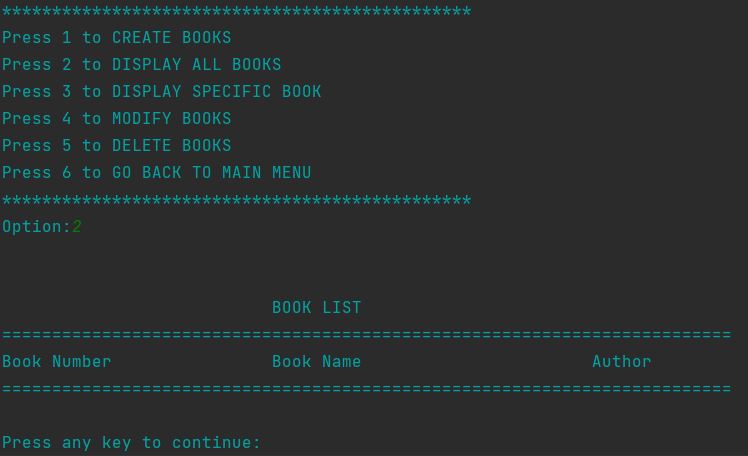
1. **Book Menu**

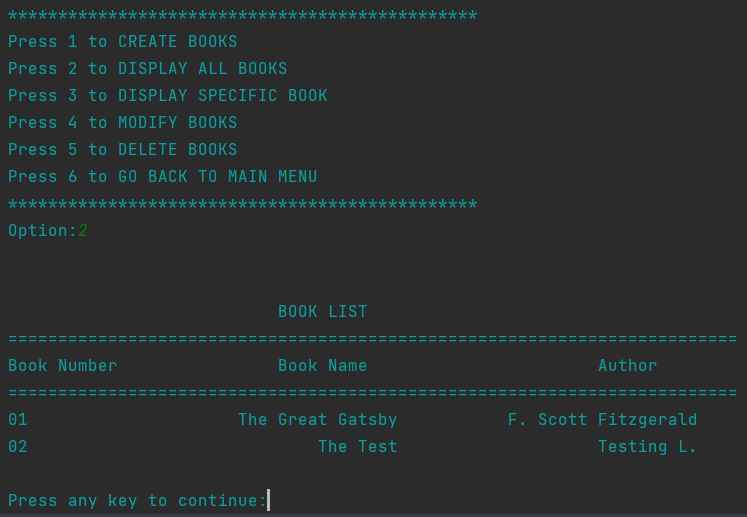
****

1. **Create Book**

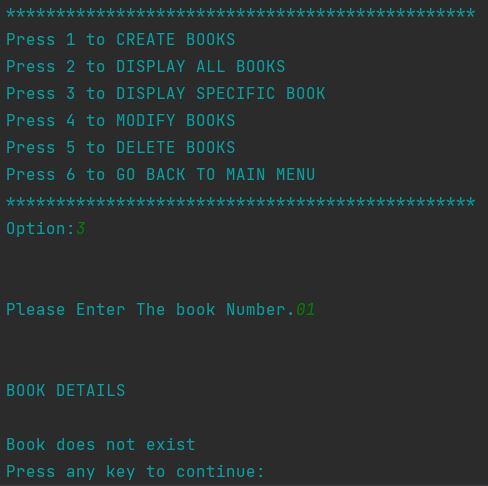
****

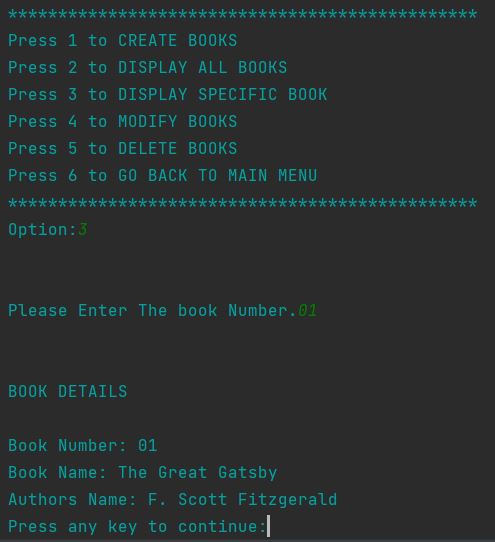
1. **Display All Books**

****

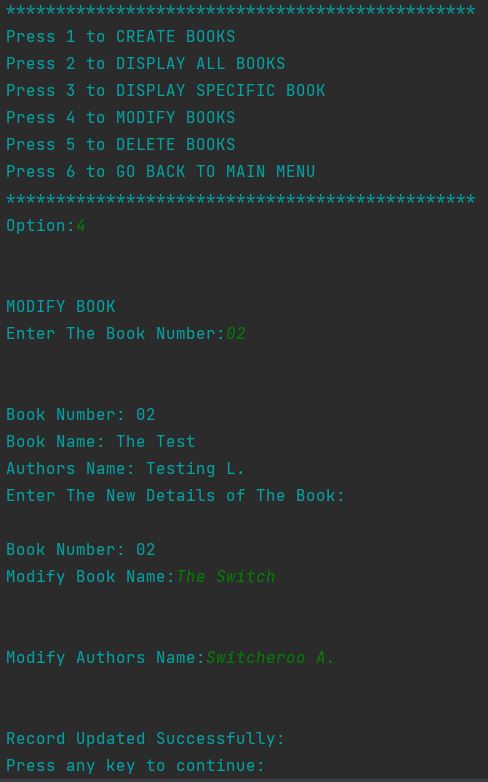
****

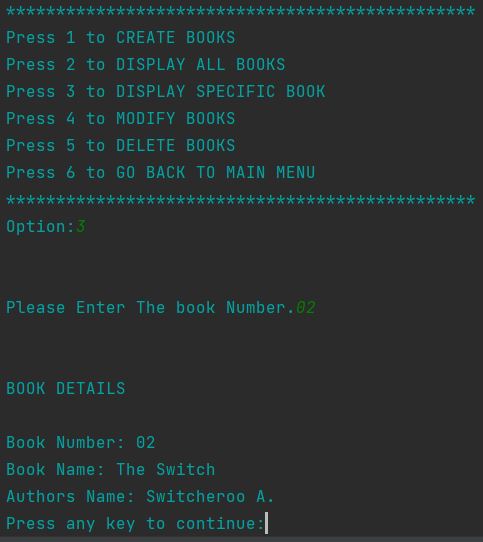
1. **Display A Book**

****

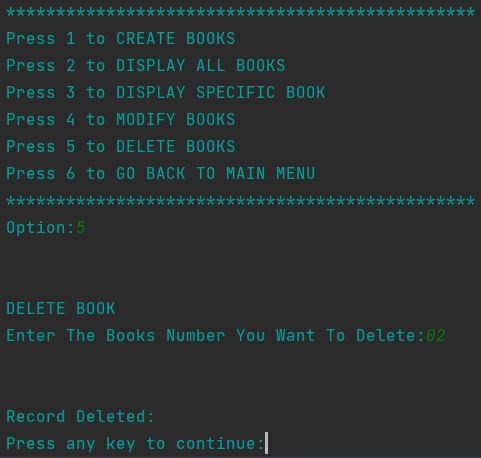
****

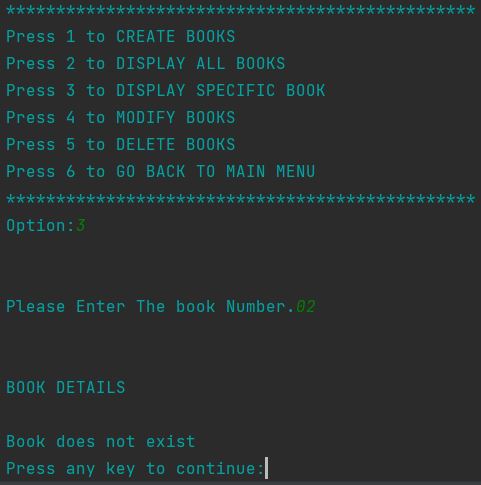
1. **Modify Book**

****

****

1. **Delete Book**

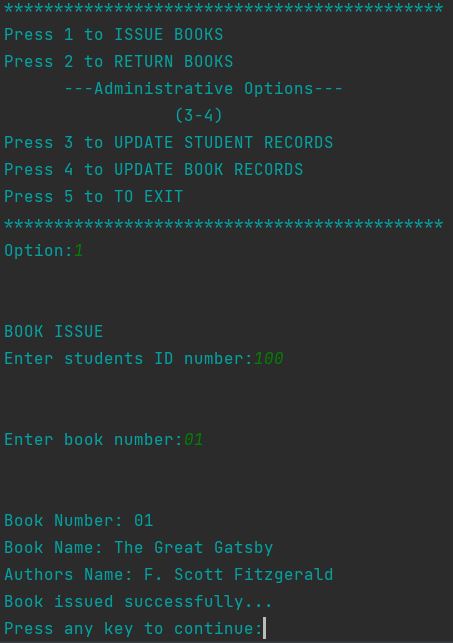
****

****

1. **Return To Main Menu**

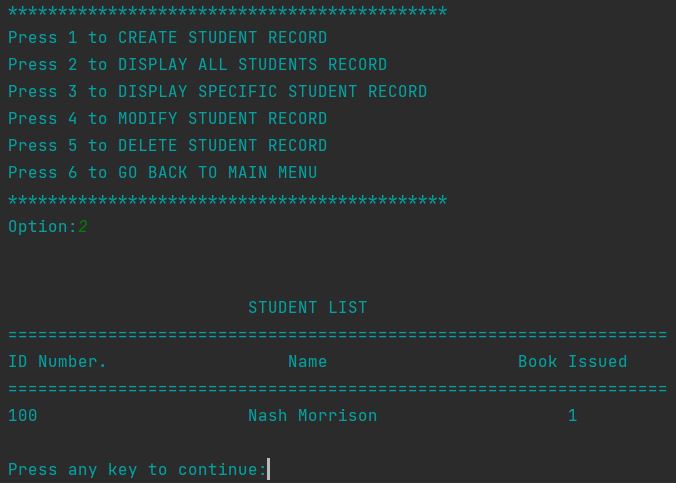
****

1. **Issue Book**

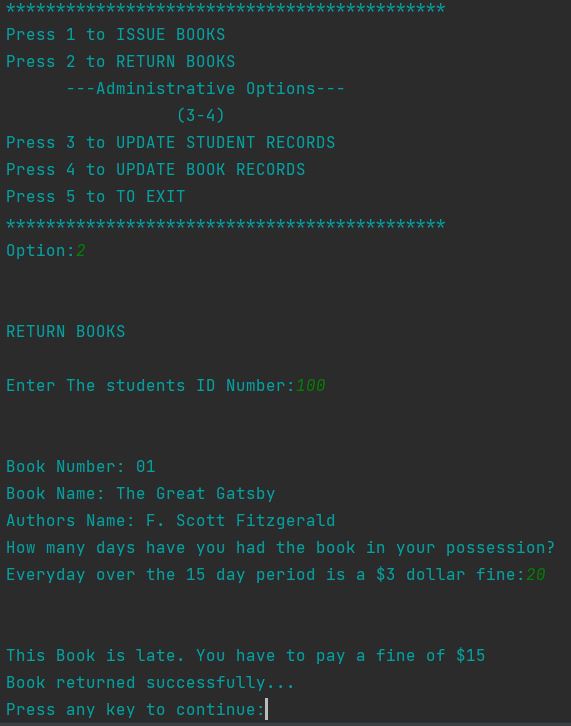
****

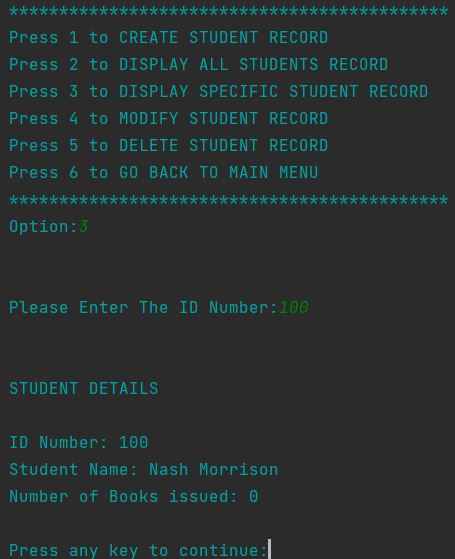
**A screen shot of a computer

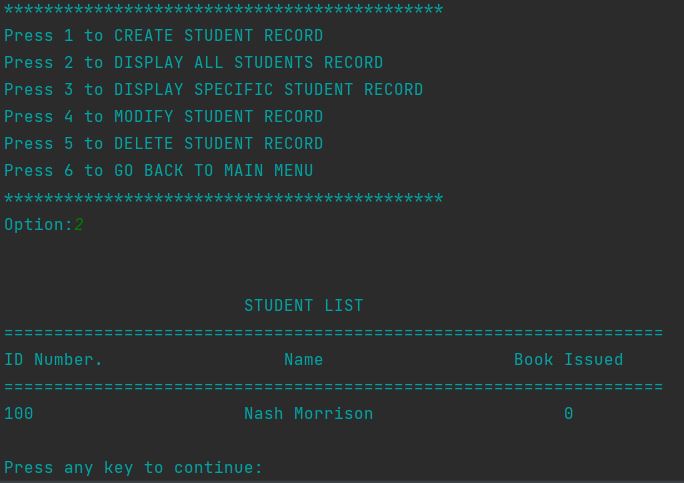
Description automatically generated**

****

1. **Return Book**

****

****

****

1. **Exit The Program**

