~\Documents\GitHub\Class-Assignments\EEL3834\Note Logger\evan_baesler_assignment5.cpp

```cpp
#include <iostream>
#include <iomanip>
#include <cstring>
#include <cctype>
#include <string>
#include <cstdlib>
#include <cmath>

using namespace std;

// Creates our class and structures
class Note { // Class that defines the structure and its variables
    public:
    string title;
    string dueDate;
    int priority;
    string completed;

    Note(){ // Default constructor, all values blank
        title = "";
        dueDate = "";
        priority = 0;
        completed = "No";
    }

    Note(string titleInput, string dueDateInput){ // Constructor that allows a title input and due date input
        title = titleInput;
        dueDate = dueDateInput;
        priority = 0;
        completed = "No";
    }

    Note(string titleInput, string dueDateInput, int priorityInput){ // Constructor that allows a title, due date, and priority input, we use this below
        title = titleInput;
        dueDate = dueDateInput;
        priority = priorityInput;
        completed = "No";
    }

};

// Adds new notes to our data
void addNote(Note notes[], int& noteCount){

    Note newNote; // Creates a blank new note using default constructor

    do {
        cout << "Enter Title: ";
        getline(cin, newNote.title); // Using getline so we can take multiple words at once (allows for spaces)

        if(newNote.title.empty()){
            cout << "Error: Title cannot be empty. \n";
        }
    } while (newNote.title.empty());

    do {
        cout << "Enter Due Date (YYYY-MM-DD): ";
        getline(cin, newNote.dueDate);
```

```
59
60          if (newNote.dueDate.length() != 10 && newNote.dueDate.length() != 11) { // I am accepting 10 or 11 here since online it said windows adds a "/r"
61              cout << "Error: Date must be in YYYY-MM-DD format.\n";              // I had issues with this but I believe our grader uses Linux
62          }
63
64      } while ((newNote.dueDate.length()) != 10 && (newNote.dueDate.length()) != 11);
65
66      do {
67          cout << "Enter Priority (1-5): ";
68          cin >> newNote.priority;
69          if (cin.fail()) {
70              cout << "Invalid input! Please enter an integer.\n";
71              cin.clear();
72              cin.ignore(1000, '\n'); // Since this uses cin, we want to make sure that getline doesn't cause an issue, so we clear it
73              return;
74          }
75          if (newNote.priority > 5 || newNote.priority < 1){
76              cout << "Error: Priority must be between 1 and 5.\n";
77          }
78      } while (1 > newNote.priority || newNote.priority > 5 || newNote.priority == 0);
79
80      notes[noteCount] = newNote; // Adds data to storage
81      noteCount++; // Updates amount of notes stored
82
83      cout << "Note added successfully!" << endl << endl;
84  }
85
86  void viewNotes(Note notes[], int noteCount){
87
88      cout << "All Notes: " << endl;
89
90      for (int i = 0; i < noteCount; i++){ // Indexes thru all notes to print information below.
91          cout << "[" << i << "] "; // Index number
92          cout << notes[i].title << " | "; // Title
93          cout << "Due: " << notes[i].dueDate << " | "; // Duedate
94          cout << "Priority: " << notes[i].priority << " | "; // Priority
95          cout << "Completed: " << notes[i].completed << endl; // Completed yes/no
96      }
97      cout << endl;
98
99  }
100
101 // Marks notes complete
102 void markComplete(Note notes[], int& noteCount){
103
104     int choice;
105
106     cout << "Enter the index of the note to mark as complete: ";
107     cin >> choice;
108
109     if (cin.fail()) { // Checks that we get an integer input
110         cout << "Invalid input! Please enter an integer.\n";
111         cin.clear();
112         cin.ignore(1000, '\n');
113         return;
114     }
115
116     notes[choice].completed = "Yes"; // Updates completion status in storage
117     cout << "Note \"" << notes[choice].title << "\" marked as completed." << endl << endl;
118
119 }
```

```
120
121 // Finds the higher priority of two notes
122 void comparePriorities(Note notes[], int& noteCount){
123
124     string input = "";
125     string output = "";
126     int index = 0;
127     int indiceOne;
128     int indiceTwo;
129
130     cout << "Enter two indices to compare (like 0 1): ";
131     getline(cin, input); // Must use getline to capture spaces/"sentences"
132
133     for(char c : input){ // For every character in the string we check for spaces
134
135         if(input[index] != ' '){ // Since we cant easily delete a character in a string, we ignore spaces
136             output += c; // We then construct a new string without the spaces
137         }
138         index += 1;
139     }
140
141     if (output.length() == 2){
142
143         indiceOne = (output[0]) - '0';
144         indiceTwo = (output[1]) - '0'; // Found online that ASCII stores char versions of integers as +48, so '5' = 53, and '0' = 48, 53 - 48 = 5
145
146         if (indiceOne >= noteCount || indiceTwo >= noteCount){  // Since subtracting the ASCII from a character should always be non-negative
147             cout << "Error: Indices out of range." << endl;     // We only check if they are too large
148             return;
149         }
150
151         else{
152             if (notes[indiceOne].priority > notes[indiceTwo].priority){
153                 cout << "\"" << notes[indiceOne].title << "\" has higher priority than \"" << notes[indiceTwo].title << "\"" << endl << endl;
154             }
155             else if (notes[indiceTwo].priority > notes[indiceOne].priority){
156                 cout << "\"" << notes[indiceTwo].title << "\" has higher priority than \"" << notes[indiceOne].title << "\"" << endl << endl;
157             }
158             else{
159                 cout << "The two indexes have the same priority.";
160             }
161         }
162
163     }
164     else{
165         cout << "Error: Input two indices!";
166     }
167 }
168
169 // Returns total notes
170 void showStats(Note notes[], int noteCount){ // Overloaded, if we do not want to show completed note count, we don't put our third boolean parameter
171     cout << "Total notes: " << (noteCount) << endl << endl; // Returns total notes
172 }
173 // Returns total completed notes
174 void showStats(Note notes[], int noteCount, bool showCompleted){ // By putting the boolean we request to show completed notes
175
176     int completedCount = 0;
177
178     for (int i = 0; i < noteCount; i++){
179         Note& note = notes[i];
180         if(note.completed == "Yes"){ // Iterates thru notes to see if this paramter is "Yes", if it is we count it
```

```
            completedCount += 1;
        }
    }

    cout << "Completed notes: " << completedCount << endl << endl;

}

// Menu function
int main()
{

    Note notes[100]; // Lets us have up to 100 notes
    int noteCount = 0; // Index variable

    while (true){
        int choice;
        int overLoadChoice;

        cout << "Note & Reminder Organizer\n";
        cout << "1. Add New Note\n";
        cout << "2. View All Notes \n";
        cout << "3. Mark Note Complete (by Reference)\n";
        cout << "4. Compare Two Notes by Priority (Friend Function)\n";
        cout << "5. Show Stats (Function Overloarding)\n";
        cout << "6. Exit\n";

        cout << "Enter your choice: ";
        cin >> choice;
        cin.ignore(1000, '\n');

        if (cin.fail()) {
            cout << "Invalid input! Please enter an integer.\n";
            cin.clear();
            cin.ignore(1000, '\n');
            continue;
        }

        cout << endl;

        if (choice == 1){
            addNote(notes, noteCount);
        }
        else if (choice == 2){
            viewNotes(notes, noteCount);
        }
        else if (choice == 3){
            markComplete(notes, noteCount);
        }
        else if (choice == 4){
            comparePriorities(notes, noteCount);
        }
        else if (choice == 5){
            cout << "Show stats: " << endl << "1. Total Notes" << endl << "2. Completed Notes" << endl << "Enter option: ";
            cin >> overLoadChoice;

            if (cin.fail()) {
                cout << "Invalid input! Please enter a valid number.\n";
                cin.clear();
                cin.ignore(1000, '\n');
            }
```

```
            else if (overLoadChoice == 1){
                showStats(notes, noteCount);
            }
            else if (overLoadChoice == 2){
                showStats(notes, noteCount, true);
            }

            else{
                cout << "Invalid input! Please enter an integer.\n";
            }


        }
        else if (choice == 6){
            cout << "Exiting program. Goodbye!";
            exit(0);
        }
    }
    return 0;
}
```

```python
## Initializes our note as a structure
class note:
    def __init__(self, title = None, dueDate = None, priority = None, completed = None):
        self.title = title
        self.dueDate = dueDate
        self.priority = priority
        self.completed = completed

## Function 1 that adds new notes to our program using a class
def addNote(notes):

    title = ""
    dueDate = ""
    priority = 0
    completed = "No"

    ## Ensures our title is not empty
    while title == "":
        title = input("Enter Title: ")
        if title == "":
            print("Error: Title cannot be empty.")

    ## Ensures the length of our due date string is 10 characters
    while len(dueDate) != 10:

        try:
            dueDate = input("Enter Due Date (YYYY-MM-DD): ")
        except:
            print("Enter Due Date (YYYY-MM-DD): ")

        if len(dueDate) != 10:
            print("Error: Date must be in YYYY-MM-DD format.")
    ## Loops until we get a non-zero priority
    while priority == 0:
        try:
            priority = int(input("Enter Priority (1-5): "))
        ## Protects against non-integer inputs
        except ValueError:
            print("Error: Priority must be between 1 and 5.")
            continue
        if priority > 5 or priority < 1:
            print("Error: Priority must be between 1 and 5.")
            priority = 0

    ## Adds our new note to our storage, adding to the end of the list
    notes.append(note(title, dueDate, priority, completed))
    print("Note added successfully!")

## Shows all data at once
def viewAllNotes(notes):

    print("All Notes: ")
    ## Enumerates is letting us index through the notes while also letting us access their data
    for i, note in enumerate(notes):
        print(f"[{i}] {note.title} | Due: {note.dueDate} | Priority: {note.priority} | Completed: {note.completed}")

## Marks the notes complete and stores the change
def markComplete(notes):

    while True:
        try:
            indexOfNote = int(input("Enter index of the note to mark complete: "))
        ## Protects against non-integer inputs
        except ValueError:
            print("Error: Index of note must be an integer.")
            continue

        ## Sets a selected note based on index number
        selectedNote = notes[indexOfNote]

        ## Checks to see if already complete, if not, it completes, if so, alerts that it is already completed
        if selectedNote.completed == "No":
            selectedNote.completed = "Yes"
            print(f"Note \"{selectedNote.title}\" marked as completed.")
            return
        elif selectedNote.completed == "Yes":
            print(f"Note \"{selectedNote.title}\" already marked as completed.")
            return
        else:
            print("Note does not exist.")
            return

## Finds the highest priority of two notes based on indices
def comparePriorities(notes):

    ## Removes spaces, splitting the characters from the spaces, defined here \/
    comparedNotes = input("Enter two indices to compare (like 0 1): ").split(" ")

    ## Checks if more than 2 indices are put in
    if len(comparedNotes) != 2:
        print("Error: Incorrect number of arguments.")

    ## Checks if the inputs were non-integers
    try:
        inputOne = int(comparedNotes[0])
        inputTwo = int(comparedNotes[1])
```

```
 97
 98         ## Makes sure the indices are not larger than our noteCount
 99         if inputOne >= len(notes) or inputTwo >= len(notes):
100             print("Error: Indices out of range.")
101             return
102
103         noteOne = notes[inputOne]
104         noteTwo = notes[inputTwo]
105
106         ## Compares respective notes
107         if inputOne > inputTwo:
108             print(f"\"{noteOne.title}\" has higher priority than \"{noteTwo.title}\".")
109
110         elif inputOne < inputTwo:
111             print(f"\"{noteTwo.title}\" has higher priority than \"{noteOne.title}\".")
112
113         else:
114             print("The two indexes have the same priority.")
115
116     except ValueError:
117         print("Error: Enter two indices to compare.")
118
119 ## Shows stats of all notes, or completed notes
120 def showStats(notes):
121
122     print("Show stats:")
123     print("1. Total Notes")
124     print("2. Compeleted Notes")
125
126     try:
127         option = int(input("Enter option: "))
128
129         if option != 1 and option != 2:
130             print("Error: Incorrect option.")
131
132         if option == 1:
133             totalNotes = len(notes)
134             print(f"Total notes: {totalNotes}")
135
136         ## Checks if notes are completed, if so, it adds one to the count and returns the sum
137         if option == 2:
138             totalCompletedNotes = 0
139             for note in notes:
140                 if note.completed == "Yes":
141                     totalCompletedNotes += 1
142             print(f"Completed notes: {totalCompletedNotes}")
143
144     except:
145         print("Error: Option must be an integer.")
146
147 ## Primary Function that calls our menu.
148 def main():
149     # Single initialization variables
150     notes = []
151     ran = False
152
153     while True:
154         choiceInt = None
155
156         if ran:
157             print()
158
159         print("Note & Reminder Organizer")
160         print("1. Add New Note")
161         print("2. View All Notes")
162         print("3. Mark Note Complete (by Reference)")
163         print("4. Compare Two Notes by Priority (Friend Function)")
164         print("5. Show Stats (Function Overloading)")
165         print("6. Exit")
166
167         ## We loop until a valid choice is selected.
168         while True:
169             try: ## Utilizes try except to ensure integer input.
170                 choiceInt = int(input("Enter your choice: "))
171                 print()
172                 if choiceInt > 6 or choiceInt < 1:
173                     print("Invalid Choice Error")
174                 else:
175                     break
176
177             except ValueError:
178                 print("Temp Error")
179
180         ran = True
181         if choiceInt == 1:
182             addNote(notes)
183         elif choiceInt == 2:
184             viewAllNotes(notes)
185         elif choiceInt == 3:
186             markComplete(notes)
187         elif choiceInt == 4:
188             comparePriorities(notes)
189         elif choiceInt == 5:
190             showStats(notes)
191         elif choiceInt == 6:
192             print("Exiting program. Goodbye!")
```

```
193             exit()
194 main()
```