Naafiul Hossain
ESE224
115107623
Tuesday 10-12:50am

Problem 1 INCLUDES p1-p3:

ShoppingList.h:

```
//Naafiul Hossian
//SBU ID: 115107623
#pragma once
#include <string>
#include <fstream>
using namespace std;
class ShoppingList {
private:
    string file_name;
    fstream myFile;
    string most_expensive_item;
    string name;
    double price;
    double max_price;
```

```
private:
    string file_name;
    fstream myFile;
    string most_expensive_item;
    string name;
    double price;
    double max_price;

public:
    ShoppingList();
    bool fileOpen(istream& in);
    bool itemExists(istream& in);
    void addItem(istream& in);
    void PrintMostExpensiveItem();
    void PrintAll();
    void PrintTranspose();
    void PrintSort();

};
```

ShoppingList.cpp

```cpp
#include "shoppingList.h"
#include <iostream>
#include <iomanip>
#include <string>
#include <cmath>
#include <vector>

using namespace std;

ShoppingList::ShoppingList() { ... }

//task1
bool ShoppingList::fileOpen(istream& in) {
    in >> file_name;
    if (file_name == "d") {
        file_name = "ItemCatalogItem.txt";
    }
    myFile.open(file_name);
    return myFile.is_open();
}
```

```cpp
void ShoppingList::PrintAll() {
    int counter = 0;
    cout << endl;
    //myFile.seekg(0, ios::beg); // Corrected the function name 'seekbeg' to 'seekg'

    while (!myFile.eof()){
        myFile >> name >> price; // Use the file read operation as the condition
        cout << ++counter << " . " << name << " is " << price << endl; // Added spaces and corrected the
    }

    //myFile.clear(); // Clear any error flags if they were set
    //myFile.seekg(0, ios::beg); // Move the file pointer back to the beginning of the file
    cout << endl;
}
```

```
void ShoppingList::PrintTranspose() {
    myFile.seekg(0, ios::beg);

    while (myFile >> name >> price) {
        int best_width = max(name.length(), to_string(price).length()) + 1;
        cout << setw(best_width) << left << name;
    }
    cout << endl;
    myFile.clear(); // Clear any error flags if they were set
    myFile.seekg(0, ios::beg);

    while (myFile >> name >> price) {
        int best_width = max(name.length(), to_string(price).length()) + 1;
        cout << setw(best_width) << left << price;
    }
    cout << endl;
    myFile.clear(); // Clear any error flags if they were set
    myFile.seekg(0, ios::beg);
}
```

```
bool ShoppingList::itemExists(istream& in) {
    string input;
    in >> input;
    myFile.seekg(0, ios::beg); // Ensure the file is at the beginning
    while (myFile >> name >> price) {
        if (name == input) {
            cout << "Item: " << name << " Price: " << price << endl;
            return true; // Item exists, return true
        }
    }
    return false;

}
```

```
void ShoppingList::addItem(istream& in) {
    string newItemName;
    double newItemPrice;

    // Read the item name and price from the user
    cout << "Enter the item name: ";
    in >> newItemName;

    cout << "Enter the item price: ";
    in >> newItemPrice;

    // Open the file in append mode to add the new item to the end
    myFile.open(file_name, ios::app);

    if (myFile.is_open()) {
        // Write the new item to the file
        myFile << newItemName << " " << newItemPrice << endl;

        cout << "Item added to the shopping list: " << newItemName << " Price: " << newItemPrice << endl;

        // Close the file to save the changes
        myFile.close();
```

```
}
void ShoppingList::PrintMostExpensiveItem() {
    // Open the file containing the shopping list data
    myFile.open(file_name);

    if (!myFile.is_open()) {
        cerr << "Error: Unable to open the file." << endl;
        return;
    }

    string mostExpensiveItemName;
    double mostExpensiveItemPrice = 0; // Initialize with a low value

    while (myFile >> name >> price) {
        // Check if the current item is more expensive than the previously found most expensive item
        if (price > mostExpensiveItemPrice) {
            mostExpensiveItemName = name;
            mostExpensiveItemPrice = price;
        }
```

```
void ShoppingList::PrintSort() {
    // Ensure the file is at the beginning
    myFile.clear();
    myFile.seekg(0, ios::beg);

    vector<pair<string, double>> Items;

    while (myFile >> name >> price) {
        Items.push_back(make_pair(name, price));
    }

    int s = Items.size();

    for (int i = 0; i < s - 1; i++) {
        bool swapped = false;

        for (int j = 0; j < s - 1 - i; j++) {
            if (Items[j].second < Items[j + 1].second) {
                swap(Items[j], Items[j + 1]);
                swapped = true;
            }
        }
```

ShoppingListManager.cpp

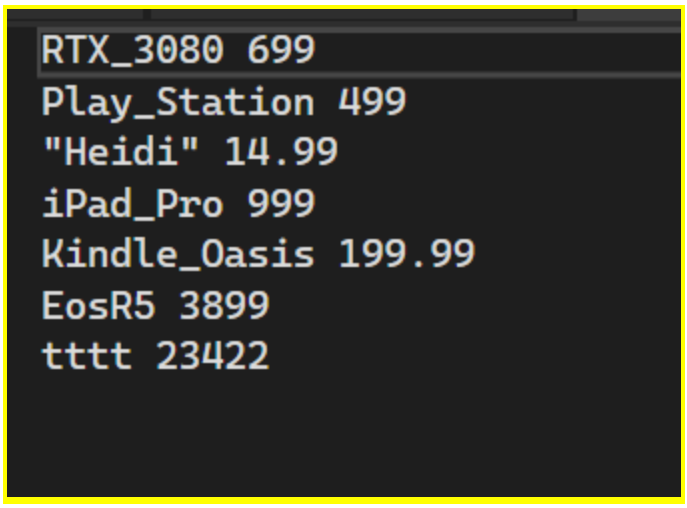
```cpp
//Naafiul Hossain
// SBU ID: 115107623

#include <iostream>
#include <fstream> // Include the <fstream> header for file operations
#include "shoppingList.h"

int main() {
    std::cout << "Opening shopping list manager... " << std::endl << std::endl;
    std::cout << "Please enter the file name: ";

    ShoppingList shoppingList;
    if (!shoppingList.fileOpen(std::cin)) {
        std::cerr << "Error opening the target file!" << std::endl;
        exit(1);
    }

    std::cout << "\nSuccessfully opening the file";
```

```cpp
    // Display a menu to the user
    while (true) {
        std::cout << " (a) - add an item and its price" << std::endl;
        std::cout << " (e) - print the most expensive item" << std::endl;
        std::cout << " (i) - check to see if this item exists" << std::endl;
        std::cout << " (p) - print all items and their prices" << std::endl;
        std::cout << " (q) - quit the program" << std::endl;
        std::cout << " (s) - print in descending order" << std::endl;
        std::cout << " (t) - print in transposed format" << std::endl;
        std::cout << "Enter your choice: ";

        char choice;
        std::cin >> choice;
```

```
        switch (choice) {
        case 'p':
            shoppingList.PrintAll();
            break;
        case 't':
            shoppingList.PrintTranspose();
            break;
        case 'e':
            shoppingList.PrintMostExpensiveItem();
            break;
        case 'i':
            std::cout << "Enter the item name to check: ";
            if (shoppingList.itemExists(std::cin)) {
                std::cout << "Item found." << std::endl;
            }
            else {
                std::cout << "Item not found." << std::endl;
            }
            break;
        case 'a':
```

```
            break;
        case 's':
            shoppingList.PrintSort();
            break;
        case 'q':
            std::cout << "Exiting the program." << std::endl;
            return 0; // Exit the program
        default:
            std::cout << "Invalid choice. Please try again." << std::endl;
        }
    }

    return 0;
}
```

ItemCatalogItem.txt:



Screenshot of the running program:

```
Opening shopping list manager...

Please enter the file name: ItemCatalogItem.txt

Successfully opening the file (a) - add an item and its price
 (e) - print the most expensive item
 (i) - check to see if this item exists
 (p) - print all items and their prices
 (q) - quit the program
 (s) - print in descending order
 (t) - print in transposed format
Enter your choice: p

1 . RTX_3080 is 699
2 . Play_Station is 499
3 . "Heidi" is 14.99
4 . iPad_Pro is 999
5 . Kindle_Oasis is 199.99
6 . EosR5 is 3899
7 . tttt is 23422

 (a) - add an item and its price
 (e) - print the most expensive item
 (i) - check to see if this item exists
 (p) - print all items and their prices
 (q) - quit the program
 (s) - print in descending order
 (t) - print in transposed format
Enter your choice: s

Prices sorted in descending order:
1. tttt $23422.00
2. EosR5        $3899.00
3. iPad_Pro     $999.00
4. RTX_3080     $699.00
5. Play_Station $499.00
```

Problem 2 with Binary Search

main.h

```cpp
// Naafiul Hossain
//SBU ID: 115107623

#include <iostream>

int searchInsertPosition(int arr[], int size, int target) {
    int left = 0;
    int right = size - 1;

    while (left <= right) {
        int mid = left + (right - left) / 2;

        if (arr[mid] == target) {
            return mid; // Target is already in the array.
        }

        if (arr[mid] < target) {
            left = mid + 1;
        }
        else {
            right = mid - 1;
        }
    }
```

```cpp
int main() {
    int n;
    std::cout << "Enter the number of elements in the sorted array: ";
    std::cin >> n;

    int* sortedArray = new int[n]; // Dynamically allocate memory for the array

    std::cout << "Enter the sorted array with distinct integer values: ";
    for (int i = 0; i < n; i++) {
        std::cin >> sortedArray[i];
    }

    int target;
    std::cout << "Enter the integer value to be inserted: ";
    std::cin >> target;
```

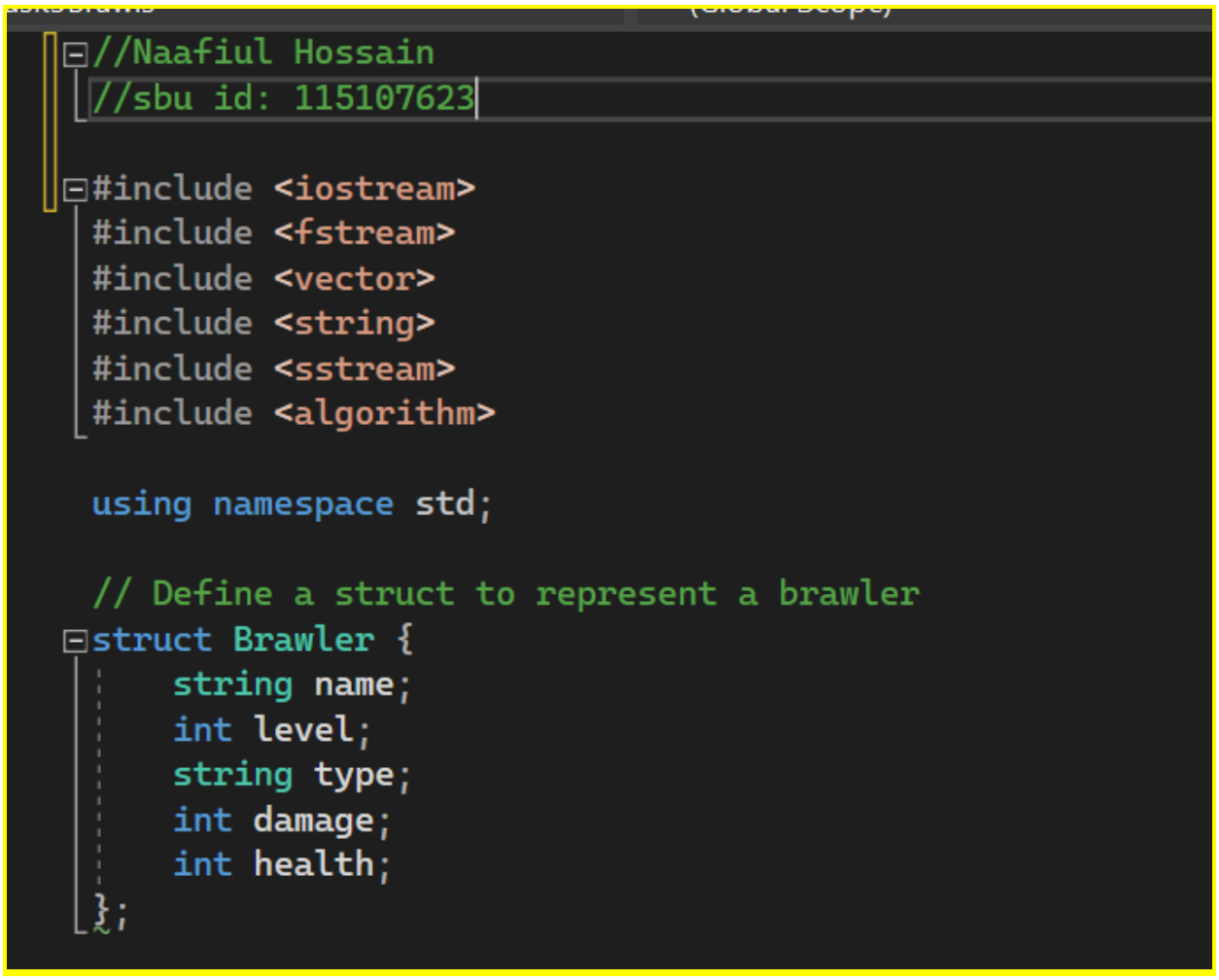
```cpp
int target;
std::cout << "Enter the integer value to be inserted: ";
std::cin >> target;

int insertIndex = searchInsertPosition(sortedArray, n, target);

std::cout << "The integer " << target << " should be inserted at index " << insertIndex << " to mai

delete[] sortedArray; // Don't forget to release the dynamically allocated memory

return 0;
```

Running of the Program:

```
Enter the number of elements in the sorted array: 5
Enter the sorted array with distinct integer values: 1 3 5 7 9
Enter the integer value to be inserted: 6
The integer 6 should be inserted at index 3 to maintain sorted order.

C:\Users\Naafiul Hossain\Documents\Lab2\Lab1Task1\NHLab5BinarySearchp4\x64\Debug\NHLa
) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debug
le when debugging stops.
Press any key to close this window . . .
```

Problem 5 BRAWLERS

Main.cpp



```cpp
//Naafiul Hossain
//sbu id: 115107623

#include <iostream>
#include <fstream>
#include <vector>
#include <string>
#include <sstream>
#include <algorithm>

using namespace std;

// Define a struct to represent a brawler
struct Brawler {
    string name;
    int level;
    string type;
    int damage;
    int health;
};
```

```cpp
int main() {
    // Create a vector to store brawlers
    vector<Brawler> brawlers;

    // Read data from the "Brawlers.dat" file
    ifstream inFile("Brawlers.dat");
    if (!inFile) {
        cerr << "Error opening the file." << endl;
        return 1;
    }

    string line;
    while (getline(inFile, line)) {
        // Split the line by comma
        stringstream ss(line);
        Brawler brawler;
        getline(ss, brawler.name, ',');
        ss >> brawler.level;
        ss.ignore(); // Ignore the comma
        getline(ss, brawler.type, ',');
```

```cpp
    inFile.close();

    // Perform part (a) by sorting the list of brawlers by their names
    sort(brawlers.begin(), brawlers.end(), [](const Brawler& a, const Brawler& b) {
        return a.name < b.name;
        });

    // Output the sorted list
    cout << "Sorted list of brawlers:" << endl;
    for (const Brawler& brawler : brawlers) {
        cout << brawler.name << " - Level: " << brawler.level << " - Type: " << brawler.type << " - Damag
    }

    // b) Find and print all the brawlers that are epic and have health above 8000
    cout << "\nEpic Brawlers with health above 8000:" << endl;
    for (const Brawler& brawler : brawlers) {
        if (brawler.type == "epic" && brawler.health > 8000) {
            cout << brawler.name << " - Health: " << brawler.health << endl;
        }
    }
```

```cpp
// c) Find and display all the brawlers that are super-rare and deal damage below 4000
cout << "\nSuper-Rare Brawlers with damage below 4000:" << endl;
for (const Brawler& brawler : brawlers) {
    if (brawler.type == "super-rare" && brawler.damage < 4000) {
        cout << brawler.name << " - Damage: " << brawler.damage << endl;
    }
}

return 0;
```

Screenshot of the running program:

```
Sorted list of brawlers:
8-Bit - Level: 9 - Type: super-rare - Damage: 7577 - Health: 2634
Ash - Level: 7 - Type: chromatic - Damage: 292 - Health: 2361
Barley - Level: 9 - Type: rare - Damage: 6637 - Health: 3858
Bea - Level: 8 - Type: epic - Damage: 1425 - Health: 8526
Belle - Level: 8 - Type: chromatic - Damage: 4543 - Health: 545
Bibi - Level: 10 - Type: epic - Damage: 2451 - Health: 8463
Bo - Level: 9 - Type: epic - Damage: 3726 - Health: 4748
Bonnie - Level: 8 - Type: epic - Damage: 6524 - Health: 4425
Brock - Level: 7 - Type: rare - Damage: 1423 - Health: 2978
Bull - Level: 7 - Type: rare - Damage: 1233 - Health: 2980
Buzz - Level: 7 - Type: chromatic - Damage: 4272 - Health: 335
Byron - Level: 7 - Type: mythic - Damage: 1424 - Health: 3935
Carl - Level: 7 - Type: super-rare - Damage: 1867 - Health: 3964
Chester - Level: 7 - Type: legendary - Damage: 2926 - Health: 373
Colt - Level: 9 - Type: rare - Damage: 6483 - Health: 8573
Darryl - Level: 10 - Type: super-rare - Damage: 2341 - Health: 3241
Dynamike - Level: 8 - Type: super-rare - Damage: 5822 - Health: 3526
Edgar - Level: 10 - Type: epic - Damage: 2545 - Health: 4564
ElPrimo - Level: 9 - Type: rare - Damage: 4637 - Health: 8363
Emz - Level: 10 - Type: epic - Damage: 3453 - Health: 6455
Eve - Level: 8 - Type: chromatic - Damage: 545 - Health: 3421
Fang - Level: 10 - Type: chromatic - Damage: 6560 - Health: 8170
Frank - Level: 8 - Type: epic - Damage: 6923 - Health: 3243
Gale - Level: 9 - Type: chromatic - Damage: 2534 - Health: 3632
Gene - Level: 5 - Type: mythic - Damage: 3347 - Health: 3978
Gray - Level: 7 - Type: mythic - Damage: 2936 - Health: 1523
Griff - Level: 8 - Type: epic - Damage: 2521 - Health: 6425
Grom - Level: 9 - Type: epic - Damage: 3628 - Health: 4738
```

```
Epic Brawlers with health above 8000:
Bea - Health: 8526
Bibi - Health: 8463

Super-Rare Brawlers with damage below 4000:
Carl - Damage: 1867
Darryl - Damage: 2341
Gus - Damage: 1947
Jacky - Damage: 2156
Penny - Damage: 495
Rico - Damage: 1426
```

BONUS

Main.cpp:

```cpp
//Naafiul Hossain
//SBU ID: 115107623

#include <iostream>
#include <vector>

using namespace std;

int search(const vector<int>& nums, int target) {
    int left = 0;
    int right = nums.size() - 1;

    while (left <= right) {
        int mid = left + (right - left) / 2;

        if (nums[mid] == target) {
            return mid;  // Found the target.
        }

        if (nums[left] <= nums[mid]) {
            // Left half is sorted.
            if (target >= nums[left] && target < nums[mid]) {
                right = mid - 1;
            }
            else {
                left = mid + 1;
            }
        }
        else {
            // Right half is sorted.
            if (target > nums[mid] && target <= nums[right]) {
                left = mid + 1;
```

```cpp
    return -1;  // Target not found in the array.
}

int main() {
    vector<int> arr1 = { 2, 3, 4, 5, 0, 1 };
    int target1 = 0;
    int result1 = search(arr1, target1);
    cout << "Output for arr1: " << result1 << endl;

    vector<int> arr2 = { 2, 4, 5, 0, 1 };
    int target2 = 3;
    int result2 = search(arr2, target2);
    cout << "Output for arr2: " << result2 << endl;

    return 0;
}
```

Program Running: