

```
Script started on 2024-09-15 13:11:43-05:00 [TERM="xterm-256color" TTY="/dev/pts/3"]
e_prevost@ares:~/Portfolio_1/Lab 2$ pwd
/home/students/e_prevost/Portfolio_1/Lab 2
e_prevost@ares:~/Portfolio_1/Lab 2$ cat seqlist.info
*****
```

Robert Prevost

CSC 122 W01

Seq List Lab

Takes in a problem set list and outputs the problems

required with the problem set name.

Base Level: Level 4

Total Level: Level 4

```
*****e_prevost@ares:~/Portfolio_1/Lab 2$ show-code driver_2.cpp
```

driver_2.cpp:

```
1 #include <iostream>
2 #include <string>
3 #include "seqlist.h"
4 #include "input_prot.h"
5 using namespace std;
6
7 int main() {
8     string psn = "Please Input your problem set's name: ";
9     string ps = "Please Input your problem set: ";
10    string pta = "Please Try Again.";
11    string programName = check_input<string>(psn, pta);
12    string numberline = check_input<string>(ps, pta);
13    string output_line = convertLineToString(programName, numberline);
14    cout << output_line;
15 }
```

```
e_prevost@ares:~/Portfolio_1/Lab 2$ showcode seqlist.h
```

showcode: command not found

```
e_prevost@ares:~/Portfolio_1/Lab 2$ show-code seqlist.h
```

seqlist.h:

```
1 #ifndef seqlist_h
2 #define seqlist_h
3 #include <string>
4 #include <vector>
5
6 std::string getStringFromTwoNumbers(char firstnum, char secondnum);
7 std::string convertLineToString(std::string lesson_name, std::string
8 problem_number_line);
9 std::vector<std::string> splitProblemNumbers(std::string
10 problem_number_line);
11 std::vector<std::string> sortAndUnique(std::vector<std::string> vec);
12 void customBubbleSort(std::vector<std::string>& vec);
13
14 #endif
```

```
e_prevost@ares:~/Portfolio_1/Lab 2$ show-code seqlist.cpp
```

seqlist.cpp:

```
1 #include <iostream>
2 #include <string>
3 #include <ctype.h>
4 #include <vector>
5 #include "seqlist.h"
6 using namespace std;
7
8 void customBubbleSort(vector<string>& vec) {
9     int n = vec.size();
10    bool swapped;
11
12    for (int i = 0; i < n - 1; i++) {
13        swapped = false;
14
15        for (int j = 0; j < n - i - 1; j++) {
16            int num1 = stoi(vec[j]);
17            int num2 = stoi(vec[j + 1]);
18
19            if (num1 > num2) {
20                swap(vec[j], vec[j + 1]);
21                swapped = true;
22            }
23        }
24
25        if (!swapped) {
26            break;
27        }
28    }
29 }
30
31 vector<string> sortAndUnique(vector<string> vec) {
32     customBubbleSort(vec);
33 }
```

```

34     vector<string> result;
35     for (string s : vec) {
36         if (result.empty() || s != result.back()) {
37             result.push_back(s);
38         }
39     }
40
41     return result;
42 }
43
44 vector<string> getStringFromTwoNumbers(string firstnum, string secondnum) {
45
46     int first = stoi(firstnum);
47     int second = stoi(secondnum);
48
49     vector<string> numStrings;
50
51     for (int i = first; i <= second; i++) {
52         numStrings.push_back(to_string(i));
53     }
54
55     return numStrings;
56 }
57
58 vector<string> splitProblemNumbers(string problem_number_line) {
59     vector<string> result;
60     string current_number;
61
62     for (char c : problem_number_line) {
63         if (isdigit(c)) {
64             current_number += c;
65         } else if (c == ',' || c == '-') {
66             if (!current_number.empty()) {
67                 result.push_back(current_number);
68                 current_number.clear();
69             }
70             result.push_back(string(1, c));
71         }
72     }
73
74     if (!current_number.empty()) {
75         result.push_back(current_number);
76     }
77
78     return result;
79 }
80
81 string convertLineToString(string lesson_name, string problem_number_line) {
82     string lesson_name_clrd;
83     for(char c : lesson_name){
84         if(c != '"' && c != '\\'){
85             lesson_name_clrd += c;
86         }
87     }

```

```

88     }
89     vector<string> number_string;
90     vector<string> problem_number_vector = splitProblemNumbers
91     (problem_number_line);
92     for(size_t i = 0; i < problem_number_vector.size(); i++){
93         if(problem_number_vector[i] != string("-") || problem_number_vector
94         [i] != string(",")){
95             if(i + 1 < problem_number_vector.size() &&
96             problem_number_vector[i+1][0] == ','){
97                 number_string.push_back(problem_number_vector[i]);
98                 i++;
99             }
100             else if(i + 2 < problem_number_vector.size() &&
101             problem_number_vector[i+1][0] == '-')
102             {
103                 vector<string> number_addition = getStringFromTwoNumbers
104                 (problem_number_vector[i],problem_number_vector[i+2]);
105
106                 for(string b: number_addition){
107                     if(!b.empty() && b.find(',') == -1){
108                         number_string.push_back(b);
109                     }
110                 }
111                 i+=2;
112             }
113             else if(!problem_number_vector[i].empty() &&
114             !problem_number_vector[i].find(',') == -1){
115                 number_string.push_back(problem_number_vector[i]);
116             }
117         }
118     }
119
120     vector<string> new_number_string = sortAndUnique(number_string);
121
122     string output_string = "Do Problem";
123     if(new_number_string.size() > 1){
124         output_string += "s";
125     }
126     output_string += " ";
127
128     for(size_t i = 0; i < new_number_string.size(); i++){
129         if(i > 0) {
130             if(i == new_number_string.size() - 1){
131                 output_string += " and ";
132             } else {
133                 output_string += ", ";
134             }
135         }
136         output_string += new_number_string[i];
137     }
138
139     output_string += " of " + lesson_name_clrd;
140     return output_string;
141

```

```
142 }
e_prevost@ares:~/Portfolio_1/Lab 2$ show-code input_prot.h
```

input_prot.h:

```
1  #ifndef input_protection_h
2  #define input_protection_h
3  #include <string>
4  #include <iostream>
5  #include <limits>
6  #include <vector>
7  template <typename varType>
8  varType check_input(std::string first_prompt, std::string try_again)
9  {
10     std::cout << first_prompt;
11     varType input;
12     std::cin >> input;
13     while (std::cin.fail()){
14         std::cerr << try_again;
15         std::cin.clear();
16         std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
17         std::cin >> input;
18     }
19     return input;
20 }
21
22
23 template <typename varType>
24 varType check_input(varType min, std::string first_prompt, std::string
25 try_again, varType max )
26 {
27     std::cout << first_prompt;
28     bool passed_min_max = false;
29     varType input;
30     std::cin >> input;
31     if(input > min && input < max && !std::cin.fail()){
32         passed_min_max = true;
33     }
34     while (std::cin.fail() || !passed_min_max){
35         std::cerr << try_again;
36         std::cin.clear();
37         std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
38         std::cin >> input;
39         if(input > min && input < max && !std::cin.fail()){
40             passed_min_max = true;
41         }
42     }
43     return input;
44 }
45
46 template <typename varType>
47 varType check_input(varType min, std::string first_prompt, std::string
```

```
48 try_again)
49 {
50     std::cout << first_prompt;
51     bool passed_min_max = false;
52     varType input;
53     std::cin >> input;
54     if(input > min && !std::cin.fail()){
55         passed_min_max = true;
56     }
57     while (std::cin.fail() || !passed_min_max){
58         std::cerr << try_again;
59         std::cin.clear();
60         std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
61         std::cin >> input;
62         if(input > min && !std::cin.fail()){
63             passed_min_max = true;
64         }
65     }
66     return input;
67 }
68
69
70 template <typename varType>
71 varType check_input( std::string first_prompt, std::string try_again,
72 varType max)
73 {
74     std::cout << first_prompt;
75     bool passed_min_max = false;
76     varType input;
77     std::cin >> input;
78     if(input < max && !std::cin.fail()){
79         passed_min_max = true;
80     }
81     while (std::cin.fail() || !passed_min_max){
82         std::cerr << try_again;
83         std::cin.clear();
84         std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
85         std::cin >> input;
86         if(input < max && !std::cin.fail()){
87             passed_min_max = true;
88         }
89     }
90     return input;
91 }
92
93 template <typename varType>
94 bool isValueInArray(const varType value, const std::vector<std::string>
95 arr)
96 {
97     std::string value_str;
98     if (std::is_same<varType, char>::value) {
99         value_str = std::string(1, value); //when char gets converted to
100         //string weird stuff happens so we try to avoid this
101     } else {
```

```

102     value_str = std::to_string(value);
103 }
104
105 for (const auto& i : arr) {
106     if (i == value_str) {
107         return true;
108     }
109 }
110 return false;
111 }
112
113 template <typename varType>
114 varType check_input(const std::vector<std::string>& string_arr, const
115 std::string& first_prompt, const std::string& try_again)
116 {
117     std::cout << first_prompt;
118     varType input;
119     while (!(std::cin >> input) || !isValueInArray(input, string_arr)) {
120         std::cerr << try_again;
121         std::cin.clear();
122         std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
123     }
124     return input;
125 }
126
127 #endif

```

e_prevost@ares:~/Portfolio_1/Lab 2\$ CPP driver_2 seqlist

driver_2.cpp***

seqlist.cpp...

seqlist.cpp: In function 'void
customBubbleSort(std::vector<std::__cxx11::basic_string<char>
>&)':

seqlist.cpp:9:21: warning: conversion
from 'std::vector<std::__cxx11::basic_string<char>
>::size_type' {aka 'long unsigned

int'} to 'int' may change value
[-Wconversion]

```

9 |     int n = vec.size();
  |           ~~~~~^

```

seqlist.cpp: In function 'std::string
convertLineToString(std::string, std::string)':
seqlist.cpp:107:50: warning:

comparison of integer expressions of different signedness:
'std::__cxx11::basic_string<char>::size_type' {aka
'long unsigned int'} and 'int'
[-Wsign-compare]

```

107 |         if(!b.empty() && b.find(',')
  |         == -1){
  |         ~~~~~^~~~~

```

seqlist.cpp:114:50: warning: logical
not is only applied to the left hand side of comparison

[-Wlogical-not-parentheses]

```

114 |         !problem_number_vector[i].find(',')

```

```

== -1){
  |
  ~~~~~^~~~~

```

seqlist.cpp:114:13: note: add
parentheses around left hand side expression to silence this warning

```

114 |         !problem_number_vector[i].find(',') == -1){
  |         ~~~~~^~~~~
  |         (
  |         )

```

seqlist.cpp:114:50: warning: comparison
of constant '-1' with boolean expression is always
false [-Wbool-compare]

```

114 |         !problem_number_vector[i].find(',')
  |         == -1){
  |         ~~~~~^~~~~

```

e_prevost@ares:~/Portfolio_1/Lab 2\$./driver_2.out

Please Input your problem set's name: 'banana

Please Input your problem set: 1-4

Do Problems 1, 2, 3 and 4 of bananae_prevost@ares:~/Portfolio_1/Lab 2\$./driver_2.out

Please Input your problem set's name: banban

Please Input your problem set: 1,20-25

Do Problems 1, 20, 21, 22, 23, 24 and 25 of banbane_prevost@ares:~/Portfolio_1/Lab 2\$

Please Input your problem set's name: "banana"

Please Input your problem set: 1-100,50-150

Do Problems 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20,

e_prevost@ares:~/Portfolio_1/Lab 2\$ ls

driver_2.cpp driver_2.out input_prot.h seqlist.cpp seqlist.h seqlist.info ser

e_prevost@ares:~/Portfolio_1/Lab 2\$ cat seqlist_tpq.txt

e_prevost@ares:~/Portfolio_1/Lab 2\$ cat seqlist_tpq.txt

1. String

2. I can use a string searching function

or manually do it myself by looping through

the string looking for a " or '.

3. Loop through the string and create

new string that contains all letters except

letters that are " or '

4. create a numeric function to

input an item into the list based on

if it is less than the item in front of it

(goes to end if no #) and greater than the

before it (goes to front if no #).

5. In this numeric insert function from problem 4

we can have a separate check for duplicates before insertion.

This separate check is a duplicate check after we sorted the

numbers from smallest to largest. The duplicate check just

then becomes looping through the problem list and checking if

the number already exists.

6. For our cout statement we will have a helper variable

that will keep track of how many chars we outputted.

once we reach 70 chars we std::endl then repeate_prevost@ares:~/Portfolio_1/Lab 2\$
exit

Script done on 2024-09-15 13:14:52-05:00 [COMMAND_EXIT_CODE="0"]