```
Script started on 2023-06-22 19:12:07-05:00 [TERM="xterm" TTY="/dev/pts/8" COLUMNS=
bj94684@ares:~$ pwd
/home/students/bj94684
bj94684@ares:~$ cat find.info
/***************************************************************************
 *                                                                        *
 *  NAME:  Jose Barron                           CLASS:  CSC122-002       *
 *                                                                        *
 *          Lab: Now where did I put that?        Level:   2              *
 *       Option:                                  Level: + 0              *
 *                                          Total Level:   2.0            *
 *                                                                        *
 * This program is designed to find a char or a string within a string.   *
 * All this information is provided by the user which is validated by the *
 * get_char or get_str from the custom library input_prot.h. After the    *
 * the program receives the information, it uses either the find_char_loc *
 * or the find_str_loc function, in the custom library strextra.h, to     *
 * find the given char or string in the given string. This custom library *
 * can be used for more than simply one string, but can be used to look   *
 * for a string in a vector/array or a list of strings.                   *
 *                                                                        *
***************************************************************************/
bj94684@ares:~$ show-code input_prot.h


input_prot.h:


#ifndef INPUT_PROT_H_INC
#define INPUT_PROT_H_INC

#include<iostream>
#include<string>
#include<limits>

using namespace std;



//This first function gets a string, and makes sure something written
// since a string can be pretty much anything but empty

inline string get_str(string prompt = "\nInput a string: ",
                    string error_msg = "\nNo string detected. Try again: "
{
    cout << prompt;
    cout.flush();
    string line;
    while ( cin.peek() == '\n')
    {
        cin.ignore();
        cout << error_msg;
        cout.flush();
    }
    getline(cin, line);

    return line;
}

//This is a simple get char function with no parameters


inline char get_ch(string prompt = "\nInput a char: ",
                   string error_msg = "\nNo char detected. Try again: ")
{
    cout << prompt;
    cout.flush();
    char t;
    while ( cin.peek() == '\n' )
    {
        cin.ignore();
        cout << error_msg;
        cout.flush();
    }
    cin >> t;
    return t;

}

//This is a more advanced get char since it forces the char to be
//one of the chars provided by the caller, and all other functions follow
//this similar function. They all force the certain data type to be one
//the caller wants. If caller wants abort, and the user aborts then
//the functions return a value that is not what the caller wants, so the
//the caller can easily identify when the user aborts

inline char get_ch_w_p(string prompt = "\nInput a char: ",
                       string error_msg = "\n Incorrect char. Try again: "
                       string parameter = "yYnN", bool abort = false)
{
    char t;
    bool done = false;
    cout << prompt;
    while ( cin.peek() == '\n' )
    {
        cin.ignore();
        cout << error_msg;
        cout.flush();
    }
    while (!done)
    {
        cin >> t;
        for (string::size_type p = 0; p != parameter.length(); ++p)
        {
            if (parameter[p] == t)
            {
                done = true;
                return t;
```

```cpp
            }
        }
        cout << error_msg;
        if (abort)
        {
            cout << "or You can abort by typing q: ";
            char a;
            cin >> a;
            if (a == 'q')
            {
                done = true;
                return t = '/';
            }
        }
    }
    return t;
}

inline double get_d_w_BR(string prompt,
                         string error_msg = "\nNumber is not in range. Try
                         double L = 0, double U = 100, bool abort = false)
{
    cout << prompt;
    cout.flush();
    double num;
    while ( cin.peek() == '\n' )
    {
        cin.ignore();
        cout << error_msg;
        cout.flush();
    }
    bool done = false;
    while ( ! done)
    {
        cin >> num;
        if ( num > L && num < U)
        {
            done = true;
            return num;
        }
        cout << error_msg;
        if (abort)
        {
            cout << "or You can abort by typing q: ";
            char a;
            cin >> a;
            if (a == 'q')
            {
                done = true;
                return num = L-1;
            }
        }
    }
    return num;
}

inline double get_d_w_LR(string prompt,
                         string error_msg = "\nNumber is not in lower end
                         double L = 0, bool abort = false)
{
    cout << prompt;
    cout.flush();
    double num;
    while ( cin.peek() == '\n' )
    {
        cin.ignore();
        cout << error_msg;
        cout.flush();
    }
    bool done = false;
    while ( ! done)
    {
        cin >> num;
        if ( num > L)
        {
            done = true;
            return num;
        }
        cout << error_msg;
        if (abort)
        {
            cout << "or You can abort by typing q: ";
            char a;
            cin >> a;
            if (a == 'q')
            {
                done = true;
                return num = L-1;
            }
        }
    }
    return num;
}

inline double get_d_w_UR(string prompt,
                         string error_msg = "\nNumber is not in upper end
                         double U = 100, bool abort = false)
{
    cout << prompt;
    cout.flush();
    double num;
    while ( cin.peek() == '\n' )
    {
        cin.ignore();
        cout << error_msg;
        cout.flush();
    }
    bool done = false;
    while ( ! done)
    {
        cin >> num;
        if (num < U)
        {
            done = true;
            return num;
        }
        cout << error_msg;
        if (abort)
        {
            cout << "or You can abort by typing q: ";
            char a;
            cin >> a;
            if (a == 'q')
            {
                done = true;
                return num = U+1;
            }
        }
    }
    return num;
}
inline long get_l_w_BR(string prompt,
                       string error_msg = "\nNumber is not in range. Try a
                       long L = 0, long U = 100, bool abort = false)
{
    cout << prompt;
    cout.flush();
    long num;
    while ( cin.peek() == '\n' )
    {
        cin.ignore();
        cout << error_msg;
        cout.flush();
    }
    bool done = false;
    while ( ! done)
    {
        cin >> num;
        if ( num > L && num < U)
        {
            done = true;
            return num;
        }
        cout << error_msg;
        if (abort)
        {
            cout << "or You can abort by typing q: ";
            char a;
            cin >> a;
            if (a == 'q')
            {
                done = true;
                return num = L-1;
            }
        }
    }
    return num;
}

inline long get_l_w_LR(string prompt,
                       string error_msg = "\nNumber is not in lower end of
                       long L = 0, bool abort = false)
{
    cout << prompt;
    cout.flush();
    long num;
    while ( cin.peek() == '\n' )
    {
        cin.ignore();
        cout << error_msg;
        cout.flush();
    }
    bool done = false;
    while ( ! done)
    {
        cin >> num;
        if ( num > L)
        {
            done = true;
            return num;
        }
        cout << error_msg;
        if (abort)
        {
            cout << "or You can abort by typing q: ";
            char a;
            cin >> a;
            if (a == 'q')
            {
                done = true;
                return num = L-1;
            }
        }
    }
    return num;
}

inline long get_l_w_UR(string prompt,
                       string error_msg = "\nNumber is not in upper end of
                       long U = 100, bool abort = false)
{
    cout << prompt;
    cout.flush();
    long num;
    while ( cin.peek() == '\n' )
    {
        cin.ignore();
        cout << error_msg;
        cout.flush();
    }
    bool done = false;
    while ( ! done)
    {
        cin >> num;
        if (num < U)
        {
            done = true;
            return num;
        }
        cout << error_msg;
        if (abort)
        {
            cout << "or You can abort by typing q: ";
            char a;
            cin >> a;
            if (a == 'q')
            {
                done = true;
                return num = U+1;
            }
        }
    }
    return num;
}



#endif
```

```
bj94684@ares:~$ show-code strextra.h


strextra.h:
```

```
#ifndef STREXTRA_H_INC
#define STREXTRA_H_INC

#include<iostream>
#include<string>
#include<vector>

using namespace std;

// Helper function that tells caller the location of a character in a
// string then stores those location in a vector. If the character is not
// in the string then the vector remains empty which helps caller
// identify when no character is in a string.
inline vector<string::size_type> num_in( string s, char t)
{
    vector<string::size_type> num;
    for (string::size_type pos = 0; pos != s.length(); ++pos)
    {
        if (static_cast<char>( s[pos] ) == t )
        {
            num.push_back(pos);
        }
    }
    return num;
}

// Helper function that tells caller if the whole string t, is in
// string s from two given locations in s and it returns a bool
// for caller to user.

inline bool incl(string s, string t, string::size_type beg,
                                     string::size_type end)
{
    string nr = s.substr(beg, end-beg+1);
    bool tf;
    if ( nr == t)
    {
        return tf = true;
    }
    return tf = false;
}

// Helper function that takes two vectors of positions in a string
// which in this function are the positions of s, then compares
// each of those combinations of positions for s to a given string
// and if they match then the location of the first vector of position.
// to tell the caller where the given string and the string match
//

inline string::size_type com(vector<string::size_type> b, vector<string::s
                 string s, string t)
{
    string::size_type num;
    for(auto p : b)
    {
        for(auto p1 : e)
        {
            if ( p > p1)    // The starter position can never be greater
            {               // than the end position
                num;
            }
            else
            {
                bool ys = incl(s, t, p, p1);
                if ( ys)
                {
                    return num = p;
                }
                num;
            }
```

```cpp
        }
    }
    return num = s.length();
}

// Helper function that takes two vectors of positions in a string
// which in this function are the positions of s, then compares
// each of those combinations of positions for s to a given string
// and each time they match then the num becomes bigger by 1. Returns
// zero if no matches occurs or returns the number of times matches
// occured.

inline short com1(vector<string::size_type> b, vector<string::size_type> e
                  string s, string t)
{
    short num;
    for(auto p : b)
    {
        for(auto p1 : e)
        {
            if ( p > p1)
            {
                num;
            }
            else
            {
                bool ys = incl(s, t, p, p1);
                if ( ys)
                {
                    ++num;
                }
                num;
            }
        }
    }
    return num;
}

// Helper function tells if string t is in string s and returns the result
// through a bool.

inline bool find_str_in(string s, string t)
{
    bool found;
    vector<string::size_type> pos = num_in(s, t[0]);
    vector<string::size_type> pos_b = num_in(s, t[ t.length() - 1]);
    if ( ! pos.empty() && ! pos_b.empty())
    {
        short ys = com1( pos, pos_b, s, t);
        if ( ys != 0)
        {
            return found = true;
        }
        else
        {
            return found = false;
        }
    }
    else
    {
        return found = false;
    }
}

// Helper function tells if string t is in string s and returns the result
// by giving the location on where it occured or if it didnt occur
// then it returns the size of string s to indicate to the caller that
// string t is not in string s.

inline string::size_type find_str_loc(string s, string t)
{
    string::size_type loc;
    vector<string::size_type> pos = num_in(s, t[0]);
    vector<string::size_type> pos_b = num_in(s, t[ t.length() - 1]);
    if ( ! pos.empty() && ! pos_b.empty())
    {
        string::size_type ys = com(pos, pos_b, s, t);
        if ( ys != s.length() )
        {
            return loc = ys;
        }
        else
        {
            return loc = s.length();
        }
    }
    else
    {
        return loc = s.length();
    }
}

inline short find_str(string s, string t)
{
    short loc = 0;
    vector<string::size_type> pos = num_in(s, t[0]);
    vector<string::size_type> pos_b = num_in(s, t[ t.length() - 1]);
    if ( ! pos.empty() && ! pos_b.empty())
    {
        string::size_type ys = com(pos, pos_b, s, t);
        if ( ys != s.length() )
        {
            ++loc;
        }
    }
    return loc;

}
```

```cpp

// Helper function that combines each possible combination of two vectors
// and returns that vector. In the future, I could template it to
// fit any data type just not strings.

inline vector<string>combine(vector<string> b, vector<string> e)
{
    vector <string> com;
    for(auto p : b)
    {
        for(auto p1 : e)
        {
            com.push_back( p + p1 );
        }
    }
    return com;
}

// Helper function that tells caller if a char is in a string returns
// results as a bool.

inline bool find_char_in( string s, char t)
{
    bool in;
    vector<string::size_type> t_in = num_in(s, t);
    if ( ! t_in.empty())
    {
        return in = true;
    }
    else
    {
        return in = false;
    }
}

// Helper function that tells the caller the location of where char t
// is in string s. Returns a vector of locations or a empty vector if
// the char was not found in t.

inline vector<string::size_type> find_char_loc( string s, char t)
{
    vector<string::size_type> in;
    vector<string::size_type> t_in = num_in(s, t);
    if ( ! t_in.empty())
    {
        for (auto p : t_in)
        {
            in.push_back(p);
        }
        return t_in;
    }
    else
    {
        return in;
    }
}

// Helper function that compares two string. Returns 1 if true
// and 0 if wrong. This function can also be templated for
// all data types not just strings and could also be a bool.

inline short compare(string s, string t)
{
    short n=0;
    if (s == t)
    {
        return ++n;
    }
    return n;
}


#endif
```

```
bj94684@ares:~$ show-code find.cpp


find.cpp:
```

```cpp
#include<iostream>
#include<string>
#include<vector>
#include<cctype>
#include<limits>
#include"input_prot.h"
#include"strextra.h"

using namespace std;

int main()
{

    cout << "\n\t\tWelcome to the Find Program\n\n";
    string L = get_str("Please input your sentence:  ",
                          "\nInvalid, try again:  ");
    string f = get_str("Please Input what you want to find:  ",
                         "\nInvalid, try again:  ");


    if (f.length() == 1)  //this means that if the string is one character
                          //which is the same as a char
    {
        char t = f[0];
        vector<string::size_type> times = find_char_loc(L,t); //searches t
        if (! times.empty() )  // if char is found then loop
        {
            bool done = false;
            bool more_than_one;
            while ( ! done )
            {

                char choice = get_ch("\nDo you want to know if your char i
                                         "Invalid, try again:  ");
                choice = static_cast<char>( toupper( choice ) );
                cin.ignore(numeric_limits<streamsize>::max(), '\n');
                if ( choice == 'Y')
                {
                    more_than_one = true; //user wants multiple locations
                    done = true;
                }
                    else if ( choice == 'N')
                {
                    more_than_one = false; // user doesnt want mul. locs.
                    done = true;
                }
                else
                {
                    cout << " You did not input any correct answer."
                         << " Please Try Again  ";
                }
            }
            if (! more_than_one) // times[0] is the first location of the
            {                    // char since it is a vector of locations
                cout << "\nYour character is in position " << times[0];
            }
            else
            {
                cout << "\nYour charcter is in positions ";
                for (vector<string::size_type>::size_type p = 0;
                     p + 1 != times.size(); ++p)
                {
                    cout << times[p] << ' ';    // goes thru all elements
                }                               // of the times vector
                cout << times.back() << '\n';   // and display all pos.
            }
        }
        else // what happens when vector times is empty
        {
            cout << "\nYour character is not in the sentence";
        }
    }
    else // if the user wants to find a string within a string
    {

       bool in = find_str_in(L,f); // is the desired string f, in the
       if(in)                      // given string L.
       {
        string::size_type loc = find_str_loc(L,f);
        cout << "The string is in the word and it is in position "
             << loc << '\n'; // list the location of the instance
       }
       else // if the desired string isnt in the given string
       {
            cout << "The string is not in the word";
       }

    }
    return 0;
}
```

```
bj94684@ares:~$ CPP find
find.cpp***
In file included from find.cpp:7:
strextra.h: In function
‘std::__cxx11::basic_string<char>::size_type
com(std::vector<long unsigned int>, std::vector<long unsigned int>,
std::string, std::string)’:
strextra.h:60:17: warning: statement
has no effect [-Wunused-value]
   60 |                 num;
      |                 ^~~
strextra.h:69:17: warning: statement
has no effect [-Wunused-value]
   69 |                 num;
      |                 ^~~
strextra.h: In function ‘short int
com1(std::vector<long unsigned int>, std::vector<long unsigned int>,
std::string, std::string)’:
strextra.h:93:17: warning: statement
has no effect [-Wunused-value]
   93 |                 num;
      |                 ^~~
strextra.h:102:17: warning: statement
has no effect [-Wunused-value]
  102 |                 num;
      |                 ^~~


bj94684@ares:~$ ./find.out

                Welcome to the Find Program

Please input your sentence:  The quick brown fox
Please Input what you want to find:  e

Do you want to know if your char is located in several locations?  no

Your character is in position 2bj94684@ares:~$ ./find.out

                Welcome to the Find Program

Please input your sentence:  the quick brown fox
Please Input what you want to find:  cow
The string is not in the wordbj94684@ares:~$ ./find.out

                Welcome to the Find Program
```

```
Please input your sentence:  11112
Please Input what you want to find:  112
The string is in the word and it is in position 2
bj94684@ares:~$ ./find
bash: ./find: No such file or directory
bj94684@ares:~$ ./find.out

                Welcome to the Find Program

Please input your sentence:  1111121212211121211212121
Please Input what you want to find:  12122
The string is in the word and it is in position 6
bj94684@ares:~$ ./find.out

                Welcome to the Find Program

Please input your sentence:  i went to store
Please Input what you want to find:  zam
The string is not in the wordbj94684@ares:~$ ./find.out

                Welcome to the Find Program

Please input your sentence:  I went to school for doing good
Please Input what you want to find:  o

Do you want to know if your char is located in several locations?  yes

Your charcter is in positions 8 13 14 18 22 28 29
bj94684@ares:~$ cat find.tpq
/***************************************************************************
*                                                                          *
* TPQs:                                                                    *
* 1. For the find char function, it takes a string and char argument.     *
* While the find string function takes two string arguments. These are    *
* different so to help the caller distinguish between them.               *
* 2. The char search function returns a vector of string::size_type while *
* the string search function returns a string::size_type data type.       *
* The char search function could find multiple locations thats why it is  *
* a vector but the string search function purpose is to find only one     *
* location because that would tell the caller that the string is found.   *
* 3. For the char search function, if it is not found then it simply      *
* returns  an empty vector and for the string search function it returns  *
* the length of the string that is being searched since it would be not   *
* possible for the string to be at that position.                         *
* 4. The compiler would distinguish two same named functions by their     *
* arguments.                                                              *
* 8. You protect the library from being circularly by including:          *
* #ifndef LIB_NAME_H_INC                                                   *
* #define LIB_NAME_H_INC                                                   *
* //                                                                       *
* #endif                                                                   *
* in the interface file.                                                   *
* 9. For the main program, In order for it to work I just #include        *
* “lib_name.h” in the top of my program since all my functions are        *
* inlined in the interface file so there is no need for an implementation *
* file.For compiling, I will need to also include the lib_name.h file.    *
* 10. My library consists solely of the interface files since all my      *
* functions are inline so they are defined there so no need for an        *
* implementation file. When using an implementation file, you need to     *
* #include the library made in this one.                                   *
*                                                                          *
***************************************************************************/
bj94684@ares:~$ exit
exit

Script done on 2023-06-22 19:16:59-05:00 [COMMAND_EXIT_CODE="0"]
```