```
Script started on 2021-10-02 19:57:36-05:00 [TERM="xterm" TTY="/dev/pts/6" COLUMNS=
k_gandhi6@ares:~$ pwd
/home/students/k_gandhi6
k_gandhi6@ares:~$ cat Hi==hi.info
Name: Kush Gandhi

Class: CSC122-001

Title: hi==Hi (Lab)



levels: 5

1.5 hi program

1.5: Add 2 arguments for skip spaces or puntuation

2: sort properly without leading 0



Description: Compares the string length by returning

values 1, -1, 0. Also, specifys any spaces skipped

or punctuation.

k_gandhi6@ares:~$ cat strextra.h
#ifndef STREXTRA_H_INC

#define STREXTRA_H_INC


#include <iostream>

#include <cctype>

#include <string>

using namespace std;


void skip(const string& skip, bool skipSpace, bool skipPunct,

    size_t& i, size_t& skippy) {//skip function to checks for skips


    while ((skipSpace && skip[i] == ' ') || (skipPunct && ispunct(skip[i]))) {

        skippy++;

        i++;

    }

}


short string_comparison(const string& str1, const string& str2,

    bool skipSpace, bool skipPunct) {

    short comp_res = 0;

    size_t skippy1{ 0 }, skippy2{ 0 }, i, j;


    for (i = 0, j = 0; i < str1.length() && j < str2.length() &&

        comp_res == 0; i++, j++) {

        //call function skip

        skip(str1, skipSpace, skipPunct, i, skippy1);

        skip(str2, skipSpace, skipPunct, j, skippy2);


        if (!isdigit(str1[i]) || !isdigit(str2[j])) {

            if (toupper(str1[i]) != toupper(str2[j])) {

                comp_res = toupper(str1[i]) < toupper(str2[j]);

            }

        }

    }

    //call to skip function

    skip(str1, skipSpace, skipPunct, i, skippy1);

    skip(str1, skipSpace, skipPunct, j, skippy2);

    //if statement to find !=, <, > or ==

    if (str1.length() - skippy1 != str2.length() - skippy2) {

       if (str1.length() - skippy1 < str2.length() - skippy2) {
```

```
            comp_res = -1;
        }
        else
            comp_res = 1;
        }
        else
            comp_res = 0;
        return comp_res;
}
#endif
k_gandhi6@ares:~$ cat Hi==hi.cpp
#include <iostream>
#include <string>
#include <cctype>
#include "strextra.h"
using namespace std;

int main() {
    bool skipSpace = false;
    bool skipPunct = false;
    string a, b, space, punct;
    short comp_res;

    cout << "Enter first string: ";
    getline(cin, a);
    cout << "Enter second string: ";
    getline(cin, b);
    cout << "\nAny Spaces Skipped: ";
    cin >> space;
    if (space[0] == 'y' || space[0] == 'Y') {
        skipSpace = true;//set bool to true
    }
    cout << "\nAny Punuation Skipped: ";
    cin >> punct;
    if (punct[0] == 'y' || punct[0] == 'Y') {
        skipPunct = true;//bool to true
    }
    //set comp_res to function to perform action
    //case switch to check the return val of function
    //and output a statement
    comp_res = string_comparison(a, b, skipSpace, skipPunct);
    switch (comp_res) {
    case -1:
        cout << "\nString a < String b" << endl;
        break;
    case 0:
        cout << "\nThey're both equal" << endl;
        break;
    case 1:
        cout << "\nstring a > string b" << endl;
        break;

    }
}
```

```
k_gandhi6@ares:~$ CPP Hi==hi
'Hi==hi.cpp'***


k_gandhi6@ares:~$ ./Hi==hi.out
Enter first string: Kush
Enter second string: kush

Any Spaces Skipped: n

Any Punuation Skipped: n

They're both equal
k_gandhi6@ares:~$ ./Hi==hi.out
Enter first string: a2
Enter second string: a10

Any Spaces Skipped: n

Any Punuation Skipped: n

String a < String b
k_gandhi6@ares:~$ ./Hi==hi.out
Enter first string: Kush
Enter second string: Kush Gandhi

Any Spaces Skipped: y

Any Punuation Skipped: n

String a < String b
k_gandhi6@ares:~$ cat Hi==hi.tpq
1. How do you compare two characters without reference to case?

How might you do this without destroying the character variable(s) contents?



    When we compare strings, we can make them const and set

    it equal to (==). The const will keep the same string

    without destroying it or being changed in mid programming.



2. How can you compare two strings in a case-insensitive way without

destroying their contents?

(You should not change the strings in order to compare them!)

    I've made a function called string_comparison() which takes

    arguments of the 2 strings by making them reference of and const.


3. What kind of arguments should your string comparison function take?

(Value, reference, constant?)


    My function takes the reference of the 2 strings and

    are const. Also, I have included bool for skips and

    punctuation which checks for true or false.


4. How do you get that weird return value for your function?

Is it always -1, 0, 1? Or is there a reason it was defined as

simply less than 0, 0, or greater than 0?


    In my function, I used if statements to compare

    the 2 strings and if they're same then return 0,

    else return -1 if smaller than or +1 for greater than.


5. How many times will you need to call your function to test it thoroughly?

How many times should you have to run the driver to do this testing?


    To test a program you would do at least 3 runs to

    check for any errors or if its working correctly.

k_gandhi6@ares:~$ exit
exit

Script done on 2021-10-02 20:07:18-05:00 [COMMAND_EXIT_CODE="0"]
```