Problem Statement:

An assembly code function, Find(Main_String, Sub_string) that finds a sub-string in a given string. This Function returns the starting position of the sub-string if found and it returns a -1 if the sub-string is not found.
This assembly function can be called in a higher programming language such as C or C++ to find a string's position in another string.

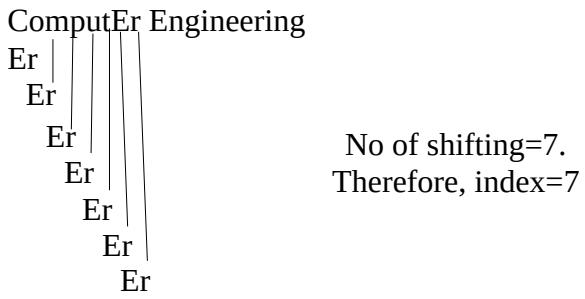
Algorithm:

We solved the problem by sliding the sub-string over the main string one by one till the sub-string is found. This condition is used **if** the length of the sub-string is less than that of the main string.

If their lengths are equal, we don't use this procedure. We just compare all the characters of the of both strings and return a 1 if they are equal and a -1 if they are not equal.

If the length of the sub-string is less than that of the main string, we return a -1 because the sub-string has more characters than the main string making it impossible for all its characters to exist in the main string.

Example:



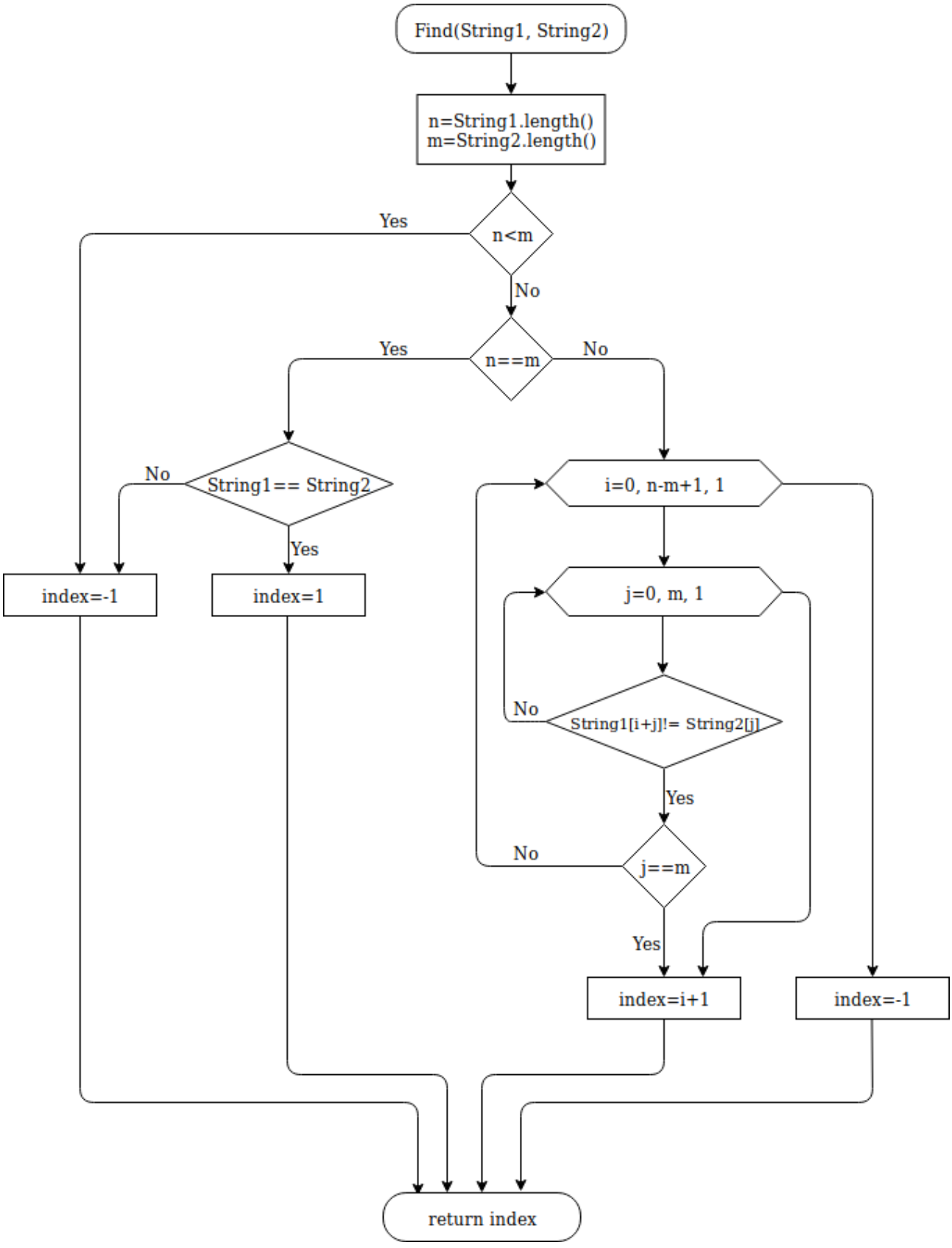
No of shifting=7.
Therefore, index=7

Algorithm Analysis:

Assuming that the length of main string is "n" and that of sub-string is "m", the algorithm takes O(m*(n-m+1)) to find the sub-string in the worst case. This is because, there are n-m+1 possible shifts of the sub-string over the main string and m character comparisons between the sub-string and main string after every shift.

The worst case O(m*(n-m+1)) is approximately $0(n^2)$ if m=n/2.

Flowchart:


Find(String1, String2)
n=String1.length()
m=String2.length()
n<m
Yes
No
n==m
Yes
No
String1== String2
No
Yes
index=-1
index=1
i=0, n-m+1, 1
j=0, m, 1
String1[i+j]!= String2[j]
No
Yes
j==m
No
Yes
index=i+1
index=-1
return index

Implementation:

C++:

```
int Find(std::string s1, std::string s2) {
        if(s2.length()> s1.length()){ return -1;}

        if(s2.length()==s1.length()){
                if(s1==s2){return 1;}
                return -1;
        }

   int i, j;
   for(i=0; i<=s1.length()-s2.length(); i++){
      for(j=0; j<s2.length(); j++){
         if(s1[i+j]!=s2[j]){ break;}
      }
      if(j==s2.length()){return i+1;}
   }
   return -1;
}
```

Assembly:

The find(string1, string2) function's assembly code can be found in the **find.asm** file.
10 labels are used alongside the find function to solve the problem using Assembly Programming Language. These labels are:

1. strLen: for calculating the length of string and substring.
2. Samelength: for computing result if the lengths are the same.
3. Find: for computing result if the length of substring is less than that of main string.
4. Check: for checking if the substring has been found.
5. Compare_Blk: for comparing each blocks.
6. Compare: for comparing each characters in block.
7. FOUND: returns position if substring is found.
8. NOTFOUND: returns -1 if substring is not found.
9. Clear: for clearing some specific registers.
10. EXIT: for returning back to find function.

Compiling The Assembly Code: **nasm -f elf64 find.asm**
Compiling Object code with C++ code: **g++ find.cpp find.o -o find**
Running Executable FIle: **./find filename**