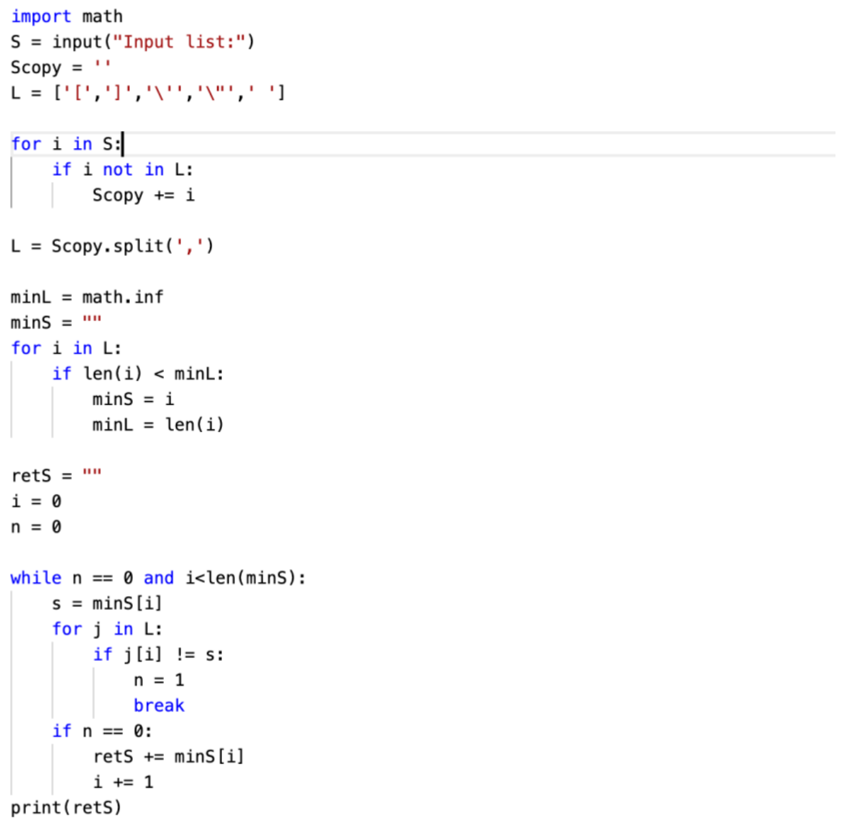
Problem 2

Step 1
1. Return string = “”
2. Find the string inside the vector with the smallest size
3. Compare the first char with the first char of rest of the strings in the vector. If they are all same, append this char to the return string and continue comparing the second char till the last char of the string with the smallest size if t; once they are different, stop comparison and return the return string.

Python version

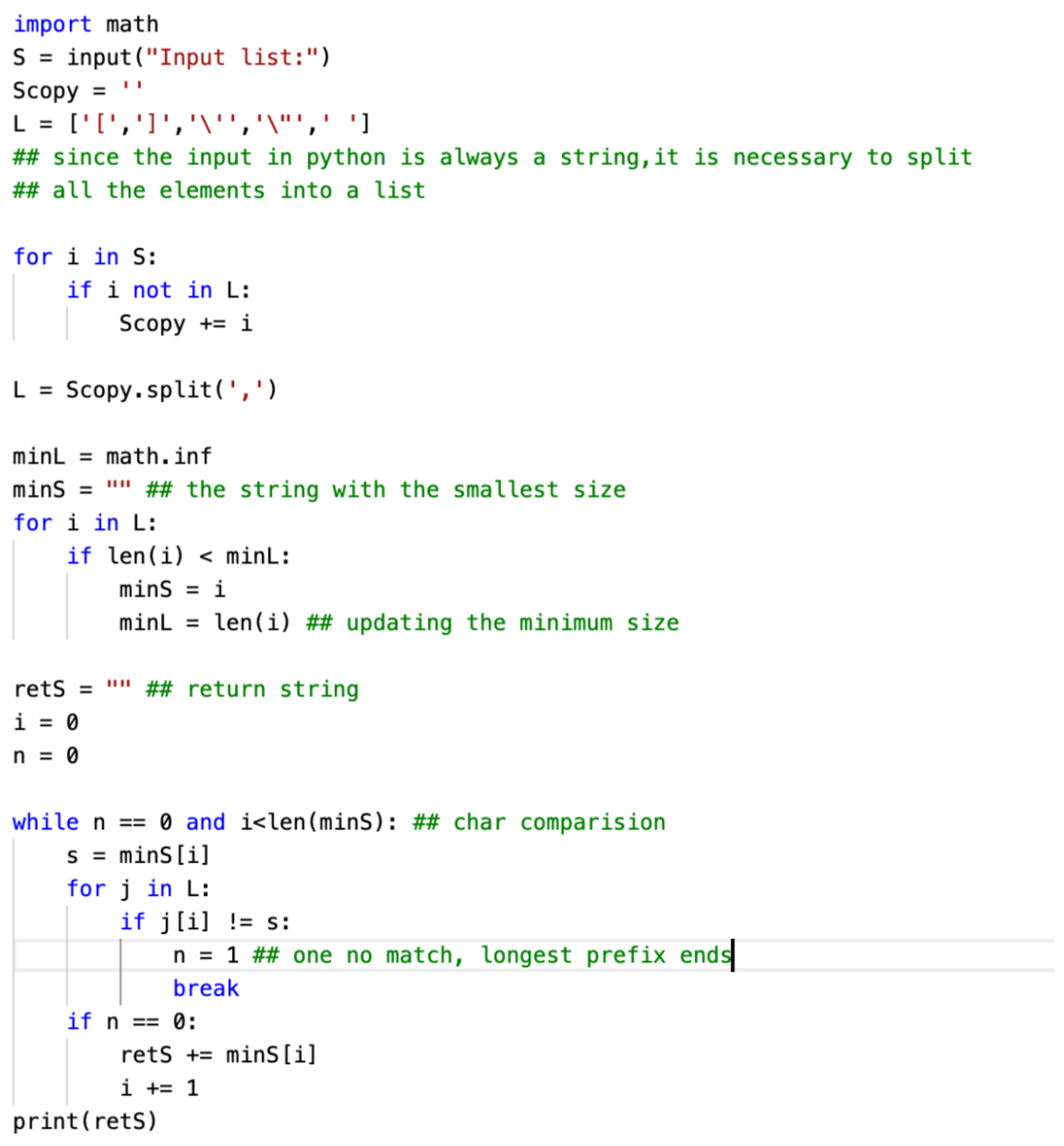
Step 2


```python
import math
S = input("Input list:")
Scopy = ''
L = ['[',']','\'','\"',' ']

for i in S:
    if i not in L:
        Scopy += i

L = Scopy.split(',')

minL = math.inf
minS = ""
for i in L:
    if len(i) < minL:
        minS = i
        minL = len(i)

retS = ""
i = 0
n = 0

while n == 0 and i<len(minS):
    s = minS[i]
    for j in L:
        if j[i] != s:
            n = 1
            break
    if n == 0:
        retS += minS[i]
        i += 1
print(retS)
```

Step 3
Same as step 2 since no syntax error

Step 4
No bugs at this point

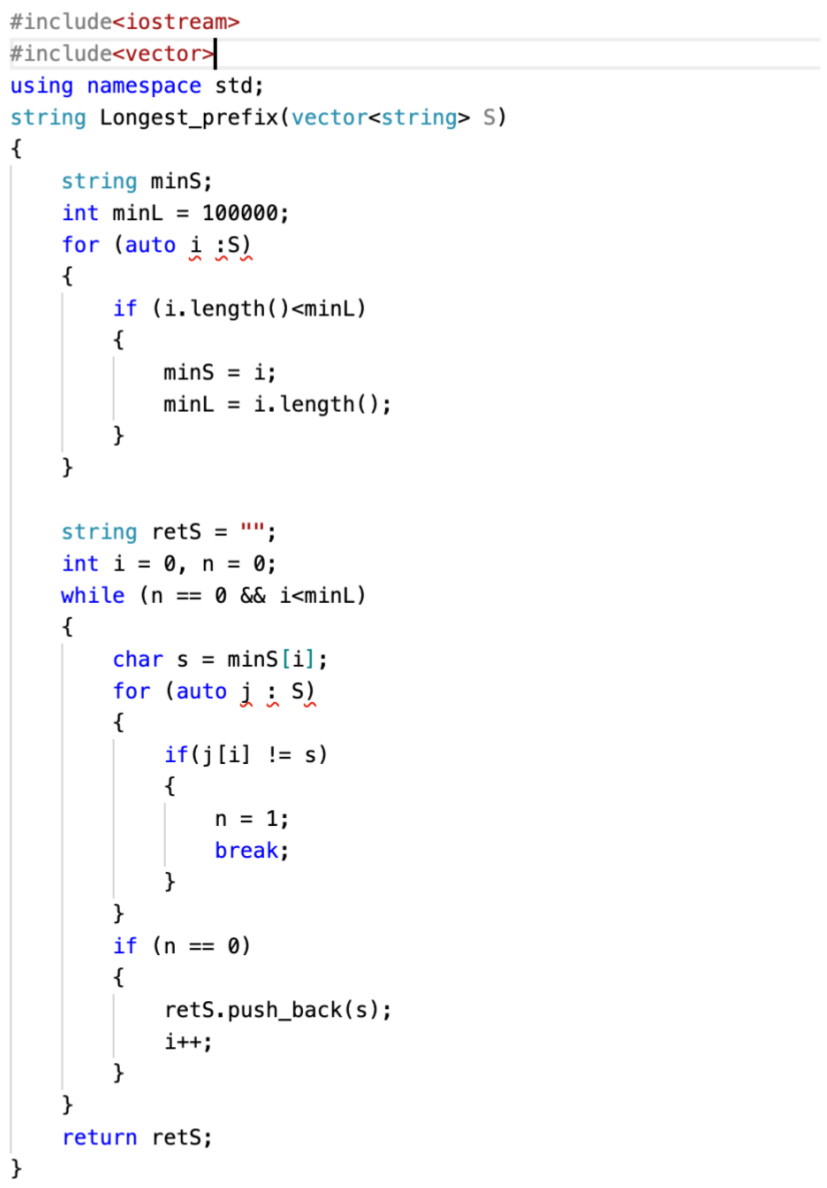
Step 5



```python
import math
S = input("Input list:")
Scopy = ''
L = ['[',']','\'','\"',' ']
## since the input in python is always a string,it is necessary to split
## all the elements into a list

for i in S:
    if i not in L:
        Scopy += i

L = Scopy.split(',')

minL = math.inf
minS = "" ## the string with the smallest size
for i in L:
    if len(i) < minL:
        minS = i
        minL = len(i) ## updating the minimum size

retS = "" ## return string
i = 0
n = 0

while n == 0 and i<len(minS): ## char comparision
    s = minS[i]
    for j in L:
        if j[i] != s:
            n = 1 ## one no match, longest prefix ends
            break
    if n == 0:
        retS += minS[i]
        i += 1
print(retS)
```

C++ version

Step 2

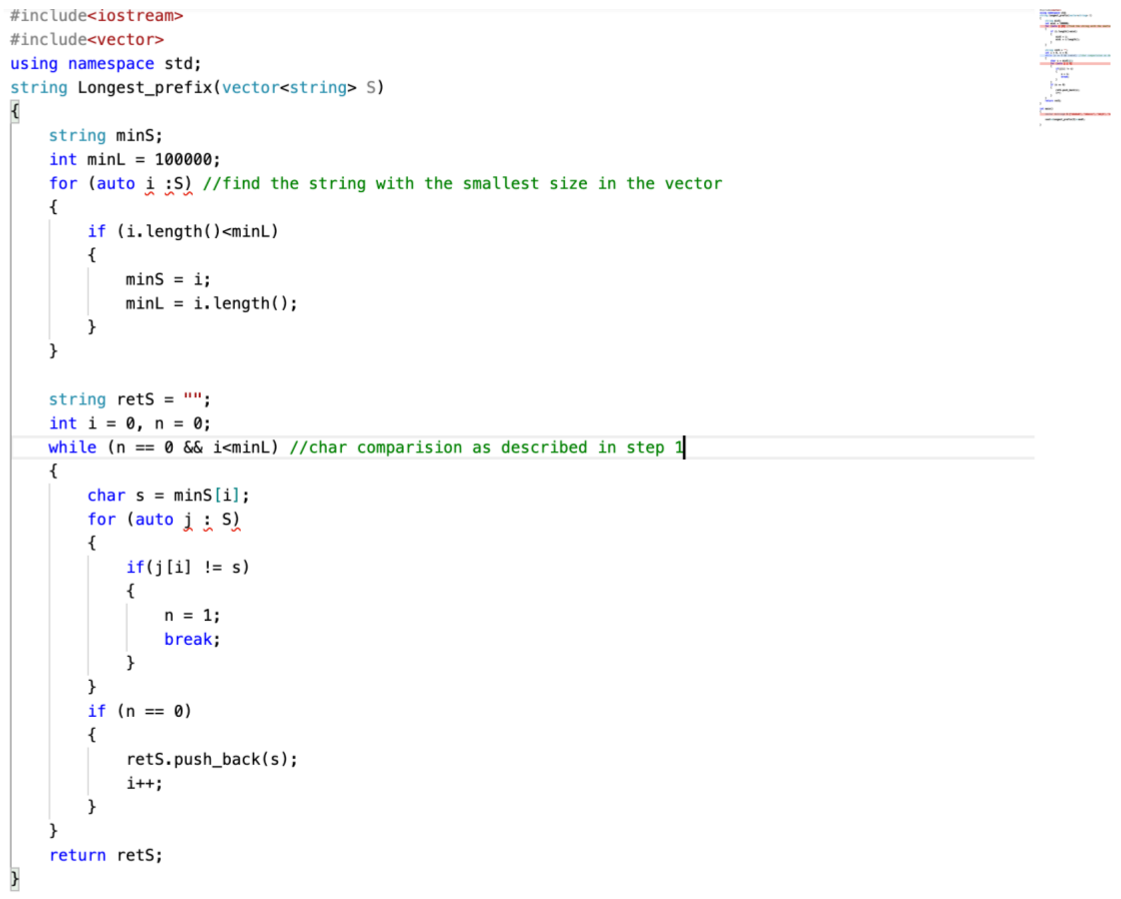


```
#include<iostream>
#include<vector>
using namespace std;
string Longest_prefix(vector<string> S)
{
    string minS;
    int minL = 100000;
    for (auto i :S)
    {
        if (i.length()<minL)
        {
            minS = i;
            minL = i.length();
        }
    }

    string retS = "";
    int i = 0, n = 0;
    while (n == 0 && i<minL)
    {
        char s = minS[i];
        for (auto j : S)
        {
            if(j[i] != s)
            {
                n = 1;
                break;
            }
        }
        if (n == 0)
        {
            retS.push_back(s);
            i++;
        }
    }
    return retS;
}
```

Step 3
Same as step 2 since there are no syntax errors

Step 4
No bugs at this point

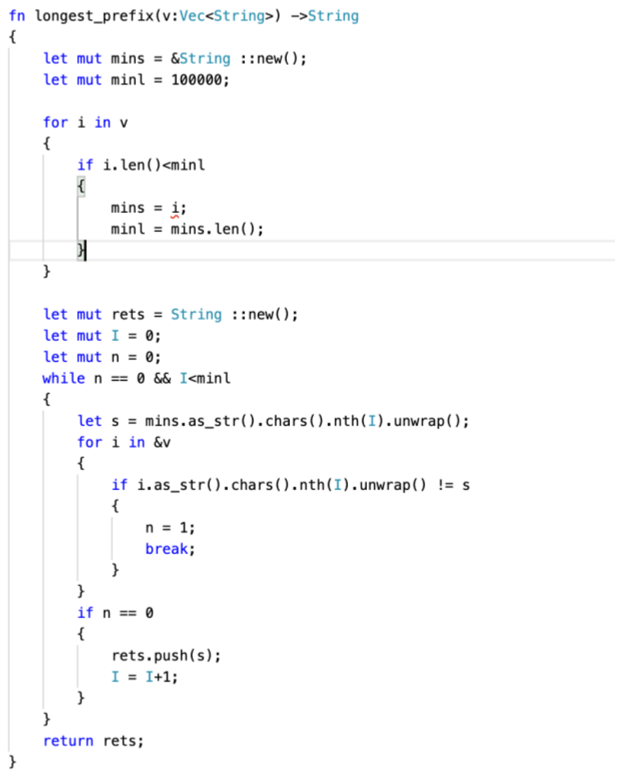
Step 5



```cpp
#include<iostream>
#include<vector>
using namespace std;
string Longest_prefix(vector<string> S)
{
    string minS;
    int minL = 100000;
    for (auto i :S) //find the string with the smallest size in the vector
    {
        if (i.length()<minL)
        {
            minS = i;
            minL = i.length();
        }
    }

    string retS = "";
    int i = 0, n = 0;
    while (n == 0 && i<minL) //char comparision as described in step 1
    {
        char s = minS[i];
        for (auto j : S)
        {
            if(j[i] != s)
            {
                n = 1;
                break;
            }
        }
        if (n == 0)
        {
            retS.push_back(s);
            i++;
        }
    }
    return retS;
}
```

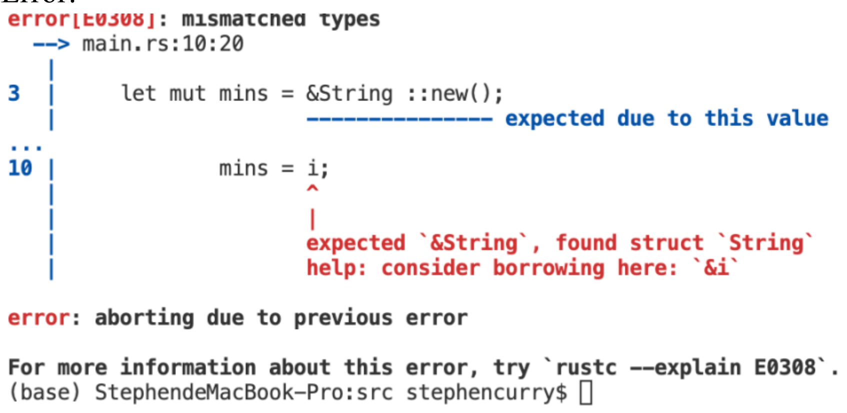
Rust version

Step 2



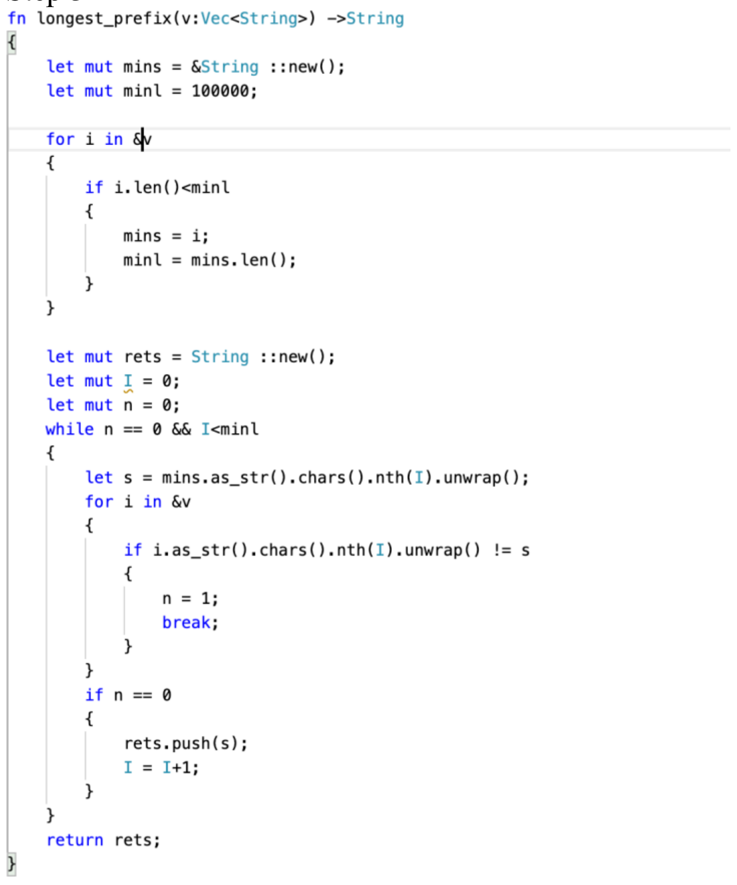
```rust
fn longest_prefix(v:Vec<String>) ->String
{
    let mut mins = &String ::new();
    let mut minl = 100000;

    for i in v
    {
        if i.len()<minl
        {
            mins = i;
            minl = mins.len();
        }
    }

    let mut rets = String ::new();
    let mut I = 0;
    let mut n = 0;
    while n == 0 && I<minl
    {
        let s = mins.as_str().chars().nth(I).unwrap();
        for i in &v
        {
            if i.as_str().chars().nth(I).unwrap() != s
            {
                n = 1;
                break;
            }
        }
        if n == 0
        {
            rets.push(s);
            I = I+1;
        }
    }
    return rets;
}
```

Error:



```
error[E0308]: mismatched types
  --> main.rs:10:20
   |
3  |     let mut mins = &String ::new();
   |                    --------------- expected due to this value
...
10 |             mins = i;
   |                    ^
   |                    |
   |                    expected `&String`, found struct `String`
   |                    help: consider borrowing here: `&i`

error: aborting due to previous error

For more information about this error, try `rustc --explain E0308`.
(base) StephendeMacBook-Pro:src stephencurry$ 
```

Step 3

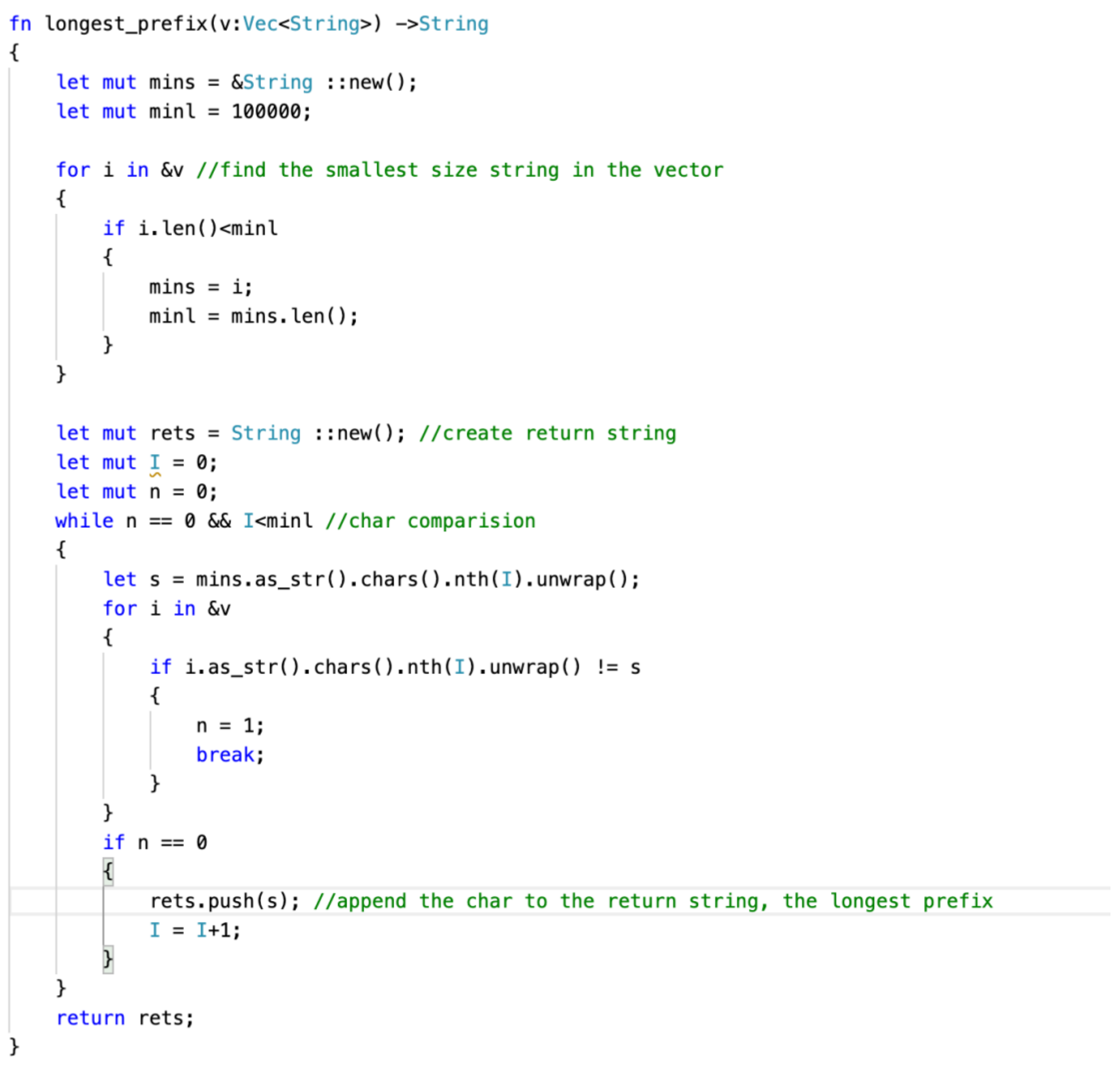


```
fn longest_prefix(v:Vec<String>) ->String
{
    let mut mins = &String ::new();
    let mut minl = 100000;

    for i in &v
    {
        if i.len()<minl
        {
            mins = i;
            minl = mins.len();
        }
    }

    let mut rets = String ::new();
    let mut I = 0;
    let mut n = 0;
    while n == 0 && I<minl
    {
        let s = mins.as_str().chars().nth(I).unwrap();
        for i in &v
        {
            if i.as_str().chars().nth(I).unwrap() != s
            {
                n = 1;
                break;
            }
        }
        if n == 0
        {
            rets.push(s);
            I = I+1;
        }
    }
    return rets;
}
```

Step 4

No bugs at this point

Step 5



```rust
fn longest_prefix(v:Vec<String>) ->String
{
    let mut mins = &String ::new();
    let mut minl = 100000;

    for i in &v //find the smallest size string in the vector
    {
        if i.len()<minl
        {
            mins = i;
            minl = mins.len();
        }
    }

    let mut rets = String ::new(); //create return string
    let mut I = 0;
    let mut n = 0;
    while n == 0 && I<minl //char comparision
    {
        let s = mins.as_str().chars().nth(I).unwrap();
        for i in &v
        {
            if i.as_str().chars().nth(I).unwrap() != s
            {
                n = 1;
                break;
            }
        }
        if n == 0
        {
            rets.push(s); //append the char to the return string, the longest prefix
            I = I+1;
        }
    }
    return rets;
}
```

Step 6 (for 3 languages)

These are all of the test cases except the given test cases in the prompt I used to test the three version of the program:

["allsdkasd", "allla", "alll", "allc"]

[“ababab", "abaccc", "abjk", "ababa"]

["banana","ban","bang","bananpick"]

["ffffff","pp","asdasda","asdasd"]