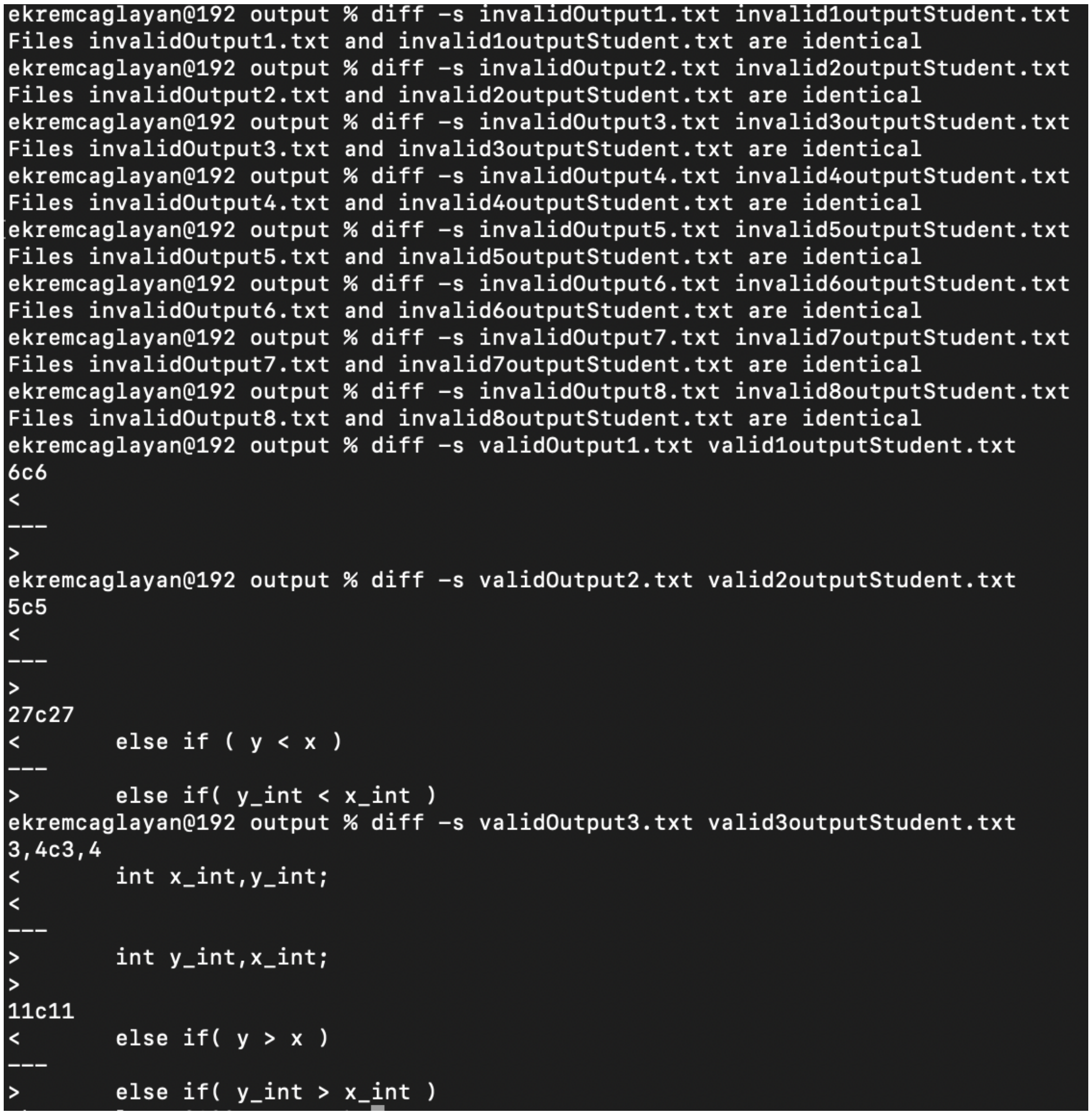
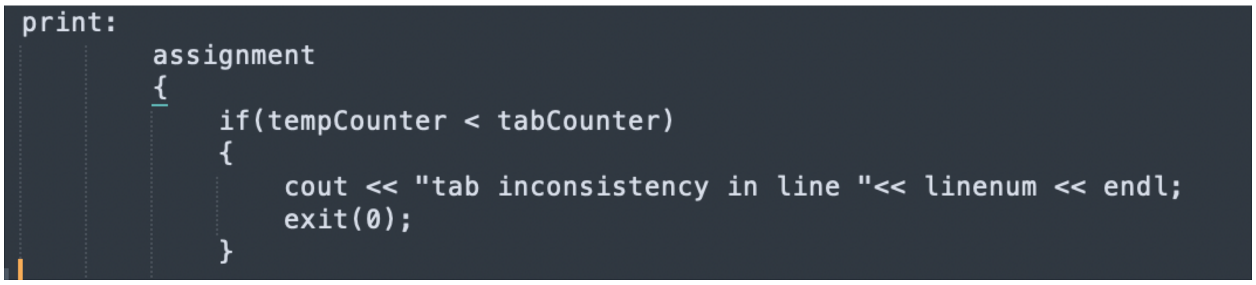


```
ekremcaglayan@192 output % diff -s invalidOutput1.txt invalid1outputStudent.txt
Files invalidOutput1.txt and invalid1outputStudent.txt are identical
ekremcaglayan@192 output % diff -s invalidOutput2.txt invalid2outputStudent.txt
Files invalidOutput2.txt and invalid2outputStudent.txt are identical
ekremcaglayan@192 output % diff -s invalidOutput3.txt invalid3outputStudent.txt
Files invalidOutput3.txt and invalid3outputStudent.txt are identical
ekremcaglayan@192 output % diff -s invalidOutput4.txt invalid4outputStudent.txt
Files invalidOutput4.txt and invalid4outputStudent.txt are identical
ekremcaglayan@192 output % diff -s invalidOutput5.txt invalid5outputStudent.txt
Files invalidOutput5.txt and invalid5outputStudent.txt are identical
ekremcaglayan@192 output % diff -s invalidOutput6.txt invalid6outputStudent.txt
Files invalidOutput6.txt and invalid6outputStudent.txt are identical
ekremcaglayan@192 output % diff -s invalidOutput7.txt invalid7outputStudent.txt
Files invalidOutput7.txt and invalid7outputStudent.txt are identical
ekremcaglayan@192 output % diff -s invalidOutput8.txt invalid8outputStudent.txt
Files invalidOutput8.txt and invalid8outputStudent.txt are identical
ekremcaglayan@192 output % diff -s validOutput1.txt valid1outputStudent.txt
6c6
<
---
>
ekremcaglayan@192 output % diff -s validOutput2.txt valid2outputStudent.txt
5c5
<
---
>
27c27
<       else if ( y < x )
---
>       else if( y_int < x_int )
ekremcaglayan@192 output % diff -s validOutput3.txt valid3outputStudent.txt
3,4c3,4
<       int x_int,y_int;
<
---
>       int y_int,x_int;
>
11c11
<       else if( y > x )
---
>       else if( y_int > x_int )
```

There is no problem in invalidOutputs. The outputs are identical as seen in the image.

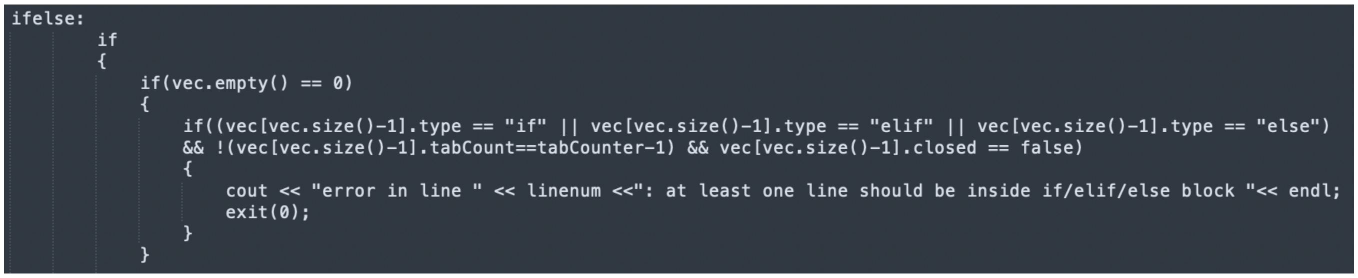
invalid1.txt :



```
print:
        assignment
        {
            if(tempCounter < tabCounter)
            {
                cout << "tab inconsistency in line "<< linenum << endl;
                exit(0);
            }
```

Tab control is requested in the assignment section of invalid1.txt, so I set one tempCounter and compared it with my tabCounter, if my previous tab count is less than the current tab count, I printed an error(“Tab inconsistency in line …”).

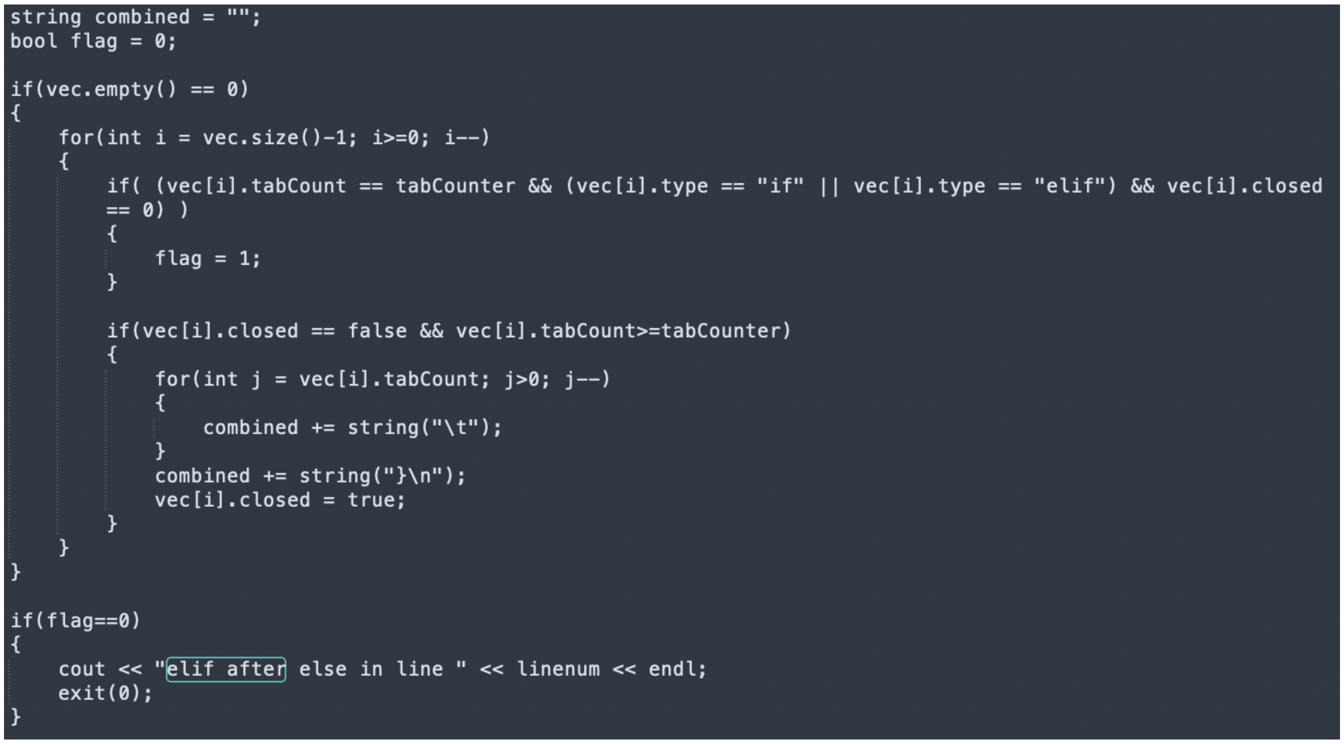
invalid2.txt - invalid3.txt - invalid5.txt :



```
ifelse:
        if
        {
            if(vec.empty() == 0)
            {
                if((vec[vec.size()-1].type == "if" || vec[vec.size()-1].type == "elif" || vec[vec.size()-1].type == "else")
                && !(vec[vec.size()-1].tabCount==tabCounter-1) && vec[vec.size()-1].closed == false)
                {
                    cout << "error in line " << linenum <<": at least one line should be inside if/elif/else block "<< endl;
                    exit(0);
                }
            }
```

In the 'ifelse' and 'assignment' sections of invalid2.txt, invalid3.txt and invalid5.txt, it is requested that an error be given if the inside of 'if', 'elif', and 'else' is empty. Therefore, if the last element of the vector does not contain a 'assignment' or 'if else elif' situation, an error will be given.

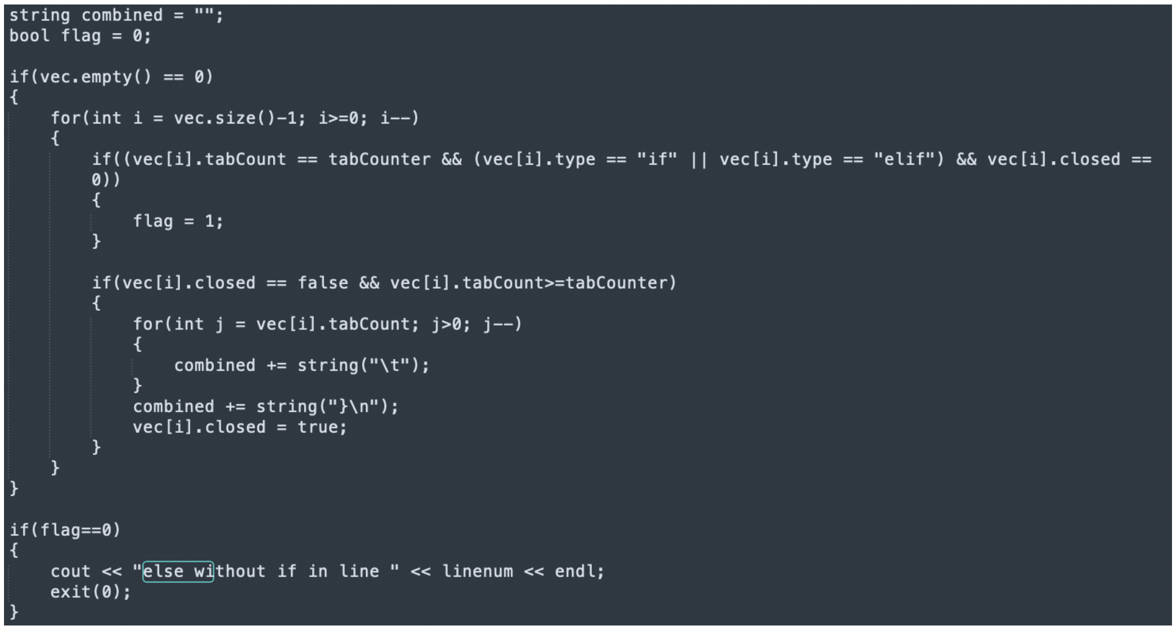
invalid4.txt



```
string combined = "";
bool flag = 0;

if(vec.empty() == 0)
{
    for(int i = vec.size()-1; i>=0; i--)
    {
        if( (vec[i].tabCount == tabCounter && (vec[i].type == "if" || vec[i].type == "elif") && vec[i].closed
        == 0) )
        {
            flag = 1;
        }

        if(vec[i].closed == false && vec[i].tabCount>=tabCounter)
        {
            for(int j = vec[i].tabCount; j>0; j--)
            {
                combined += string("\t");
            }
            combined += string("}\n");
            vec[i].closed = true;
        }
    }
}

if(flag==0)
{
    cout << "elif after else in line " << linenum << endl;
    exit(0);
}
```

In the 'ifelse' section of invalid4.txt, it is requested that an error be given if 'elif' comes after 'else'. Therefore, the last element added to the vector is checked. If the number of tabs is the same as the previous number of tabs, and there is only an 'if' and 'elif' before 'elif', no error is given. However, an error(“elif after else in line … ”) is given if there is an 'else'.

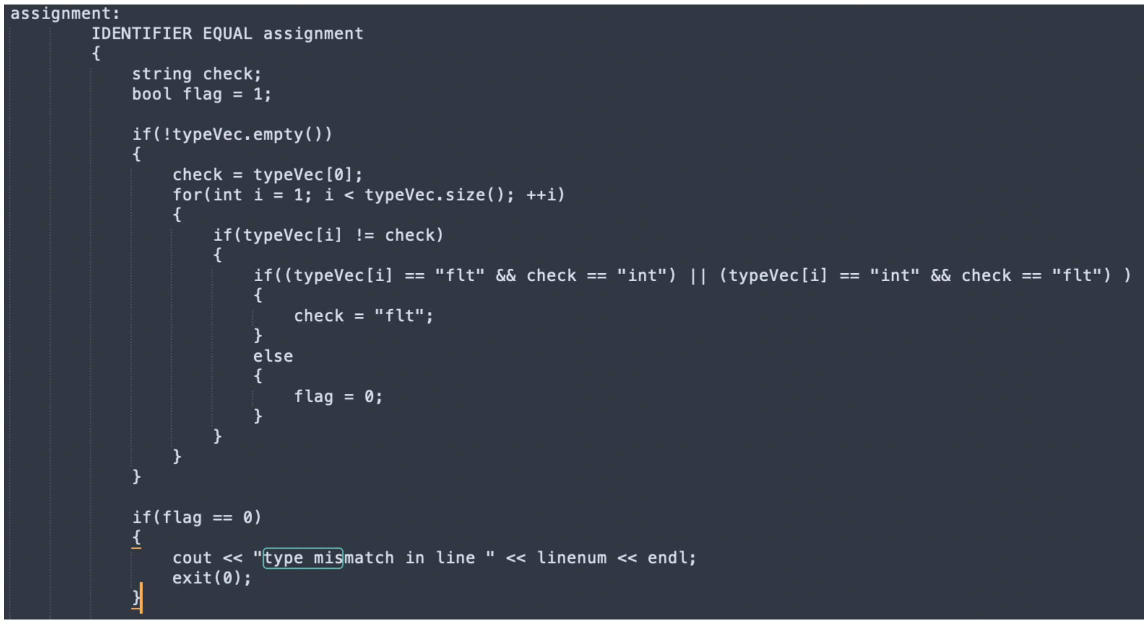
invalid6.txt :



```
string combined = "";
bool flag = 0;

if(vec.empty() == 0)
{
    for(int i = vec.size()-1; i>=0; i--)
    {
        if((vec[i].tabCount == tabCounter && (vec[i].type == "if" || vec[i].type == "elif") && vec[i].closed ==
        0))
        {
            flag = 1;
        }

        if(vec[i].closed == false && vec[i].tabCount>=tabCounter)
        {
            for(int j = vec[i].tabCount; j>0; j--)
            {
                combined += string("\t");
            }
            combined += string("}\n");
            vec[i].closed = true;
        }
    }
}

if(flag==0)
{
    cout << "else without if in line " << linenum << endl;
    exit(0);
}
```

In the 'ifelse' section of invalid6.txt, it is requested that an error be given if there is no 'if' before 'else'. Therefore, the last element added to the vector is checked. If the number of tabs is the same as the previous number of tabs, and there is no 'if' before 'else', an error(“else without if in line”) is given.

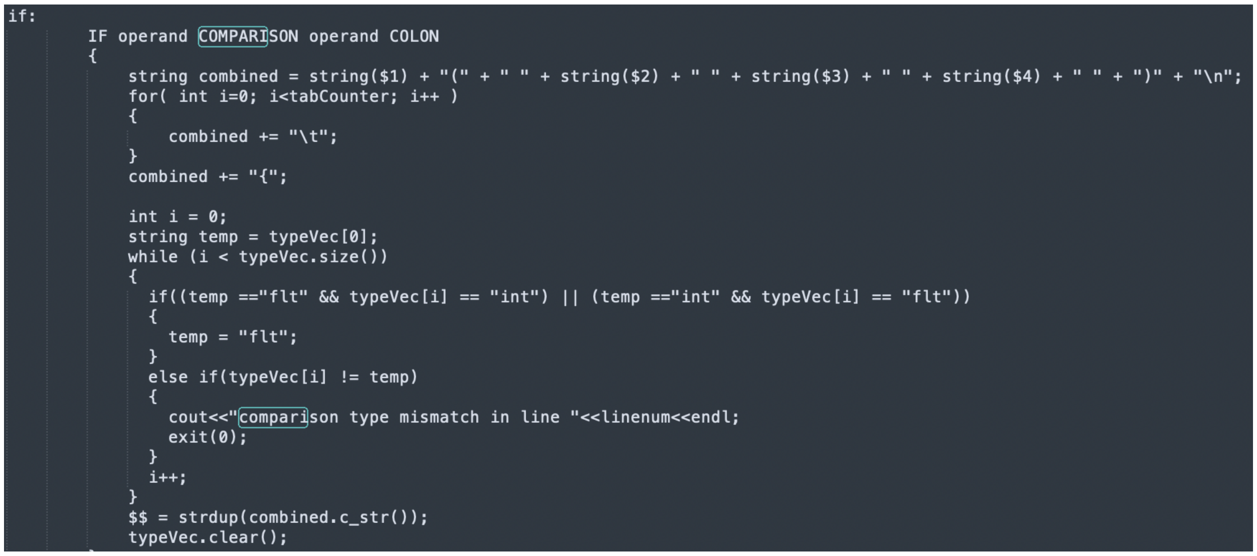
invalid7.txt :



```
assignment:
        IDENTIFIER EQUAL assignment
        {
            string check;
            bool flag = 1;

            if(!typeVec.empty())
            {
                check = typeVec[0];
                for(int i = 1; i < typeVec.size(); ++i)
                {
                    if(typeVec[i] != check)
                    {
                        if((typeVec[i] == "flt" && check == "int") || (typeVec[i] == "int" && check == "flt") )
                        {
                            check = "flt";
                        }
                        else
                        {
                            flag = 0;
                        }
                    }
                }
            }

            if(flag == 0)
            {
                cout << "type mismatch in line " << linenum << endl;
                exit(0);
            }
```

It is desired that an error be given in the invalid7.txt when we try to perform arithmetic operations with ‘strings’ and ‘int-float’ in the assignment section. Therefore, if we perform arithmetic operations in the form of ‘float’ and ‘int’ or ‘int’ and ‘float’, it returns a ‘float’ value, but if it is in the form of ‘float-int’ and ‘string’ or ‘string’ and ‘float-int’, it gives an error(type mismatch).

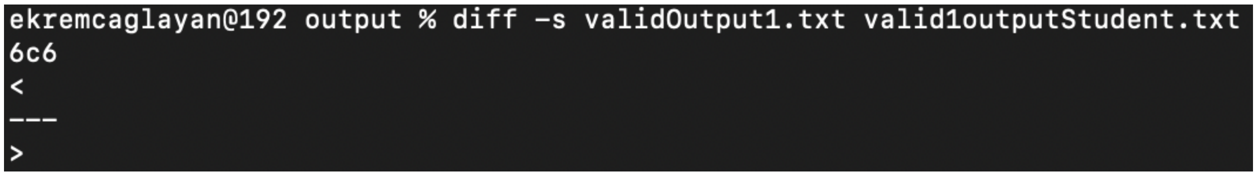
invalid8.txt :



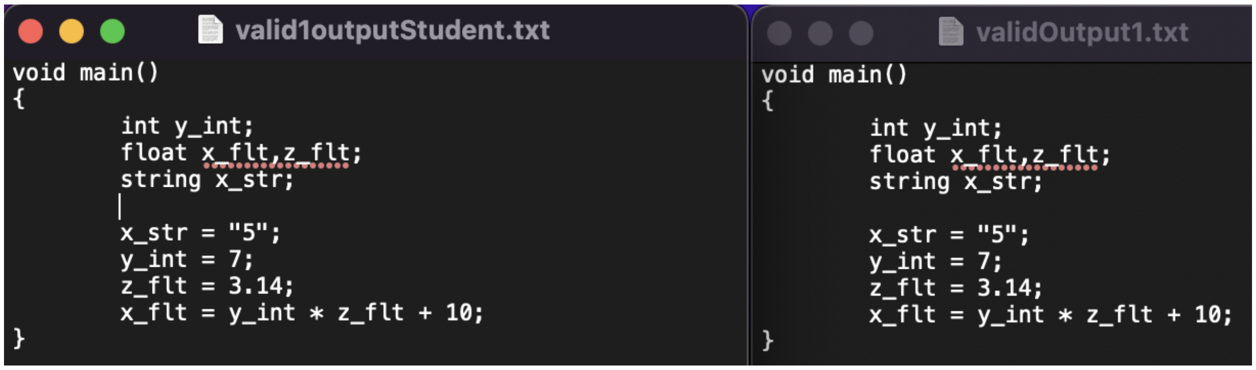
```
if:
        IF operand COMPARISON operand COLON
        {
            string combined = string($1) + "(" + " " + string($2) + " " + string($3) + " " + string($4) + " " + ")" + "\n";
            for( int i=0; i<tabCounter; i++ )
            {
                combined += "\t";
            }
            combined += "{";

            int i = 0;
            string temp = typeVec[0];
            while (i < typeVec.size())
            {
              if((temp =="flt" && typeVec[i] == "int") || (temp =="int" && typeVec[i] == "flt"))
              {
                temp = "flt";
              }
              else if(typeVec[i] != temp)
              {
                cout<<"comparison type mismatch in line "<<linenum<<endl;
                exit(0);
              }
              i++;
            }
            $$ = strdup(combined.c_str());
            typeVec.clear();
```

It is desired that an error be given in the invalid8.txt when we try to comparison operations with ‘strings’ and ‘int-float’ in the ifelse section. Therefore, if we perform comparison operations in the form of ‘float’ and ‘int’ or ‘int’ and ‘float’, it returns a ‘float’ value, but if it is in the form of ‘float-int’ and ‘string’ or ‘string’ and ‘float-int’, it gives an error(comparison type mismatch).

valid1.txt :



```
ekremcaglayan@192 output % diff -s validOutput1.txt valid1outputStudent.txt
6c6
<
---
>
```



valid1outputStudent.txt

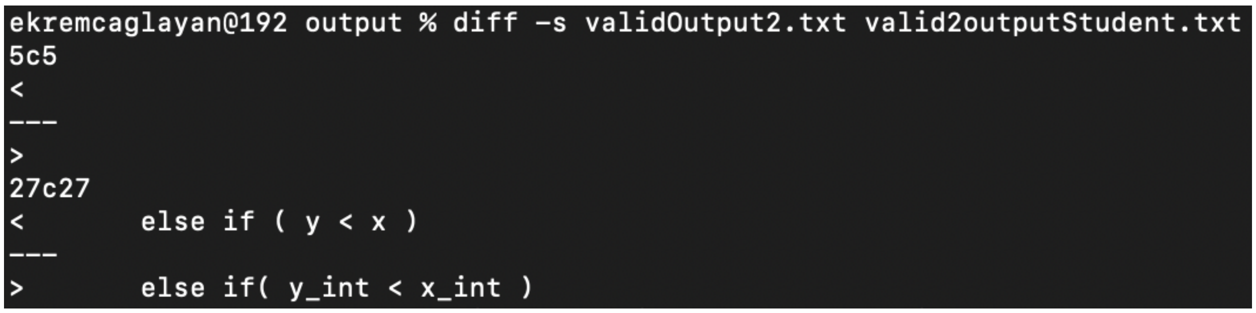
```
void main()
{
        int y_int;
        float x_flt,z_flt;
        string x_str;
        
        x_str = "5";
        y_int = 7;
        z_flt = 3.14;
        x_flt = y_int * z_flt + 10;
}
```

validOutput1.txt

```
void main()
{
        int y_int;
        float x_flt,z_flt;
        string x_str;

        x_str = "5";
        y_int = 7;
        z_flt = 3.14;
        x_flt = y_int * z_flt + 10;
}
```

Here, every time I make a newline, I add a tab, so it sees the tab as a difference, as shown in the image. Except for that, there is no other error in my output.

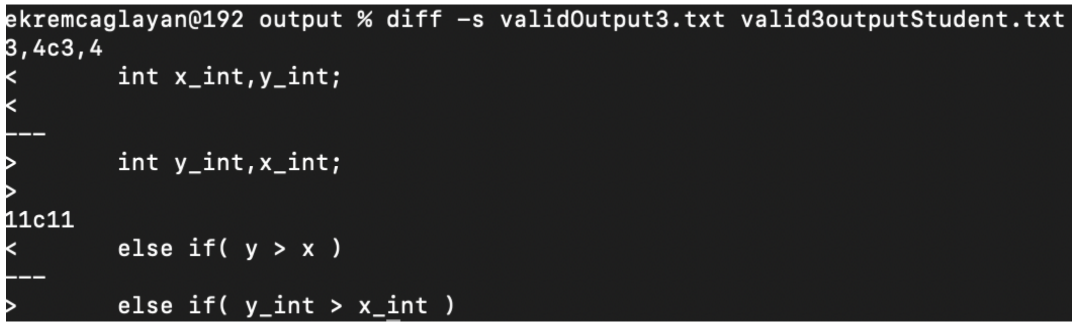
valid2.txt :



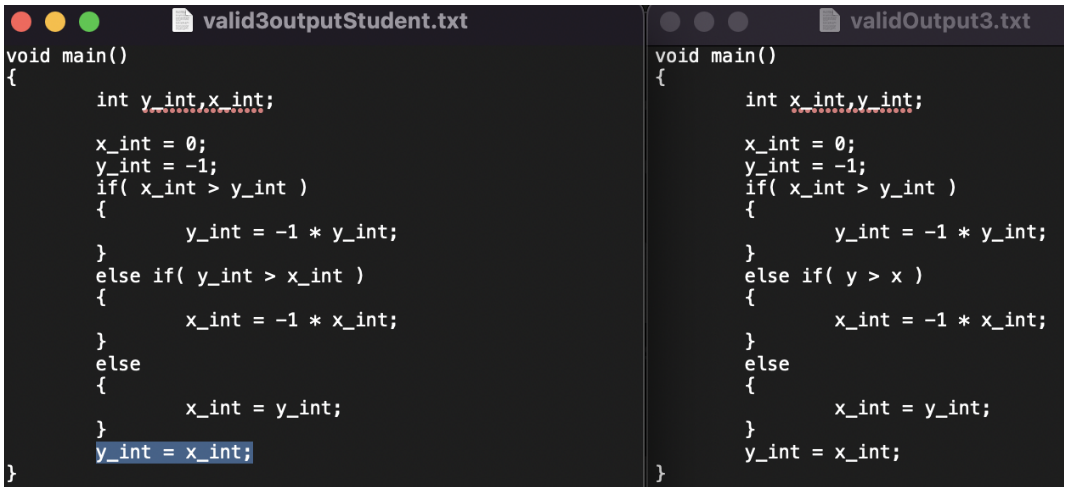
```
ekremcaglayan@192 output % diff -s validOutput2.txt valid2outputStudent.txt
5c5
<
---
>
27c27
<       else if ( y < x )
---
>       else if( y_int < x_int )
```

There is again ‘tab’ problem and I am getting an error in the ‘else if’ section because you have not specified the types inside it. However, I think this is something you have missed. I also added type checks inside the ‘else if’ and printed them out, just like we did in the ‘if’.

valid3.txt :



```
ekremcaglayan@192 output % diff -s validOutput3.txt valid3outputStudent.txt
3,4c3,4
<       int x_int,y_int;
<
---
>       int y_int,x_int;
>
11c11
<       else if( y > x )
---
>       else if( y_int > x_int )
```



```
void main()
{
        int y_int,x_int;

        x_int = 0;
        y_int = -1;
        if( x_int > y_int )
        {
                y_int = -1 * y_int;
        }
        else if( y_int > x_int )
        {
                x_int = -1 * x_int;
        }
        else
        {
                x_int = y_int;
        }
        y_int = x_int;
}
```

```
void main()
{
        int x_int,y_int;

        x_int = 0;
        y_int = -1;
        if( x_int > y_int )
        {
                y_int = -1 * y_int;
        }
        else if( y > x )
        {
                x_int = -1 * x_int;
        }
        else
        {
                x_int = y_int;
        }
        y_int = x_int;
}
```

There is again ‘tab’ and ‘else if’ problem here and I am defining my values with insert at the top, so when the value of that value changes, it reinserts and takes it back to the beginning, which is why there is a difference in the sorting.