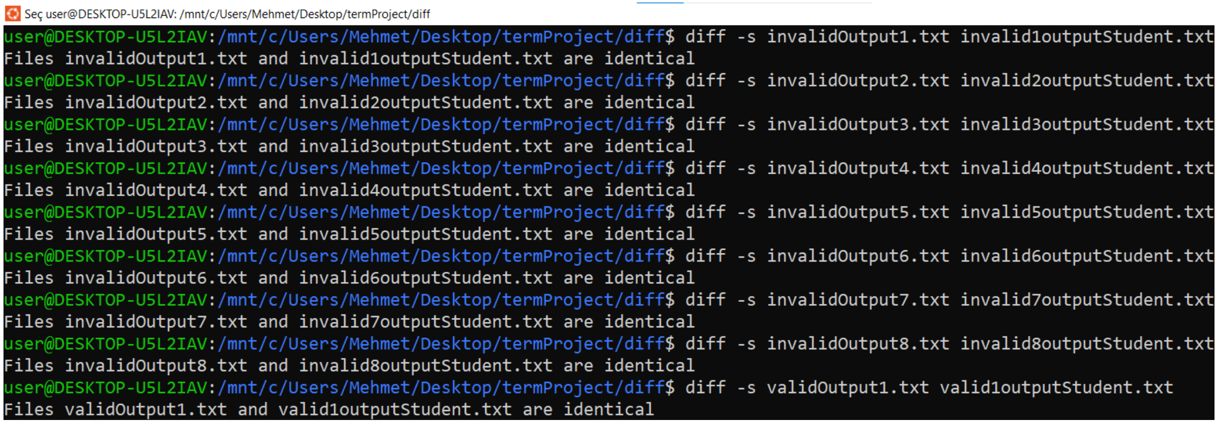
Mehmet Şemdin Aktay 20190702102 08.01.2023

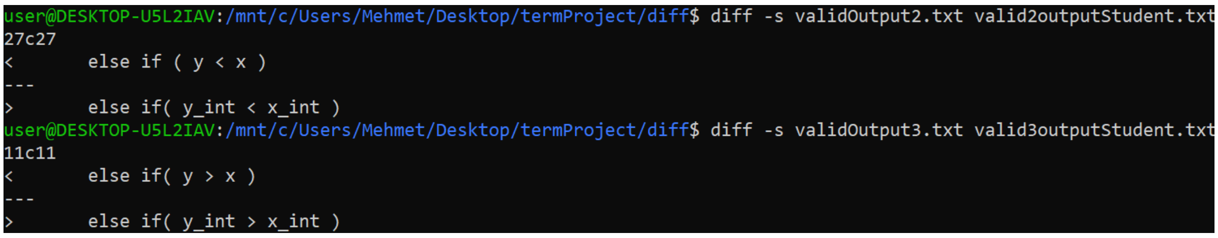
CSE 351 TERM PROJECT REPORT

When I tried all of the ipnuts.txt, I got “identical” results for 8 invalid inputs and for 1 valid input.



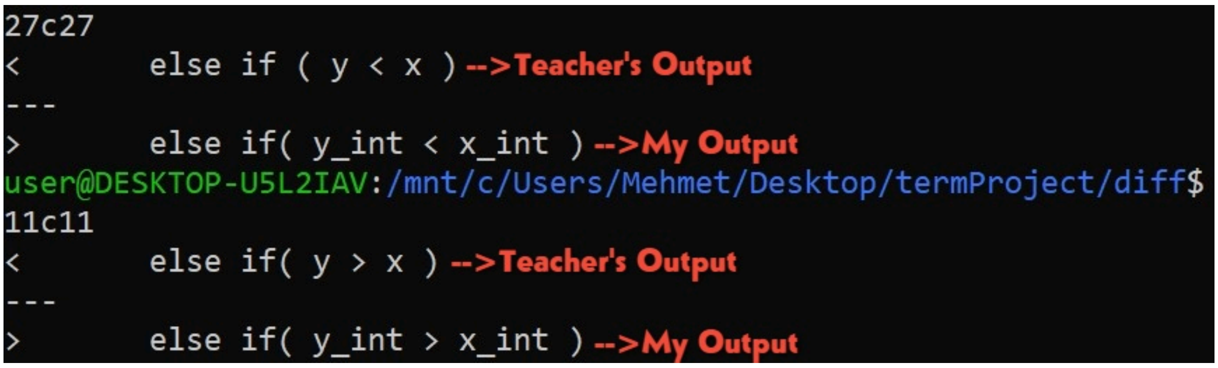
```
Seç user@DESKTOP-U5L2IAV: /mnt/c/Users/Mehmet/Desktop/termProject/diff
user@DESKTOP-U5L2IAV:/mnt/c/Users/Mehmet/Desktop/termProject/diff$ diff -s invalidOutput1.txt invalid1outputStudent.txt
Files invalidOutput1.txt and invalid1outputStudent.txt are identical
user@DESKTOP-U5L2IAV:/mnt/c/Users/Mehmet/Desktop/termProject/diff$ diff -s invalidOutput2.txt invalid2outputStudent.txt
Files invalidOutput2.txt and invalid2outputStudent.txt are identical
user@DESKTOP-U5L2IAV:/mnt/c/Users/Mehmet/Desktop/termProject/diff$ diff -s invalidOutput3.txt invalid3outputStudent.txt
Files invalidOutput3.txt and invalid3outputStudent.txt are identical
user@DESKTOP-U5L2IAV:/mnt/c/Users/Mehmet/Desktop/termProject/diff$ diff -s invalidOutput4.txt invalid4outputStudent.txt
Files invalidOutput4.txt and invalid4outputStudent.txt are identical
user@DESKTOP-U5L2IAV:/mnt/c/Users/Mehmet/Desktop/termProject/diff$ diff -s invalidOutput5.txt invalid5outputStudent.txt
Files invalidOutput5.txt and invalid5outputStudent.txt are identical
user@DESKTOP-U5L2IAV:/mnt/c/Users/Mehmet/Desktop/termProject/diff$ diff -s invalidOutput6.txt invalid6outputStudent.txt
Files invalidOutput6.txt and invalid6outputStudent.txt are identical
user@DESKTOP-U5L2IAV:/mnt/c/Users/Mehmet/Desktop/termProject/diff$ diff -s invalidOutput7.txt invalid7outputStudent.txt
Files invalidOutput7.txt and invalid7outputStudent.txt are identical
user@DESKTOP-U5L2IAV:/mnt/c/Users/Mehmet/Desktop/termProject/diff$ diff -s invalidOutput8.txt invalid8outputStudent.txt
Files invalidOutput8.txt and invalid8outputStudent.txt are identical
user@DESKTOP-U5L2IAV:/mnt/c/Users/Mehmet/Desktop/termProject/diff$ diff -s validOutput1.txt valid1outputStudent.txt
Files validOutput1.txt and valid1outputStudent.txt are identical
```

As you see above picture, All the invalid inputs are identical and one valid input is identical. For the other two valid inputs. As you see in the picture below. I just got same difference.



```
user@DESKTOP-U5L2IAV:/mnt/c/Users/Mehmet/Desktop/termProject/diff$ diff -s validOutput2.txt valid2outputStudent.txt
27c27
<        else if ( y < x )
---
>        else if( y_int < x_int )
user@DESKTOP-U5L2IAV:/mnt/c/Users/Mehmet/Desktop/termProject/diff$ diff -s validOutput3.txt valid3outputStudent.txt
11c11
<        else if( y > x )
---
>        else if( y_int > x_int )
```

This is because the output that I am given is not true. In “else if” block it is not written type of variables such as “y” and “x”. As you see my output below, “y” and “x” are “y_int” and “x_int”.

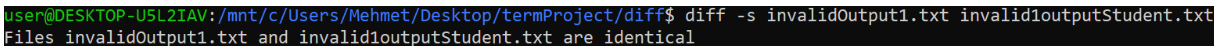


```
27c27
<        else if ( y < x ) -->Teacher's Output
---
>        else if( y_int < x_int ) -->My Output
user@DESKTOP-U5L2IAV:/mnt/c/Users/Mehmet/Desktop/termProject/diff$
11c11
<        else if( y > x ) -->Teacher's Output
---
>        else if( y_int > x_int ) -->My Output
```

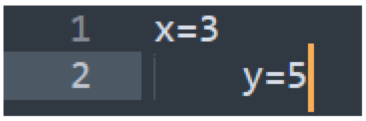
It is same as for **valid3output.txt.**

For all “invalid Identical” parts:

1) InvalidOutput1.txt



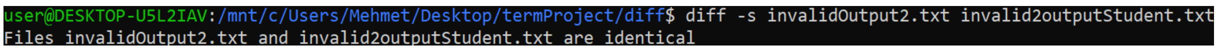
```
user@DESKTOP-U5L2IAV:/mnt/c/Users/Mehmet/Desktop/termProject/diff$ diff -s invalidOutput1.txt invalid1outputStudent.txt
Files invalidOutput1.txt and invalid1outputStudent.txt are identical
```



```
x=3
	y=5
```

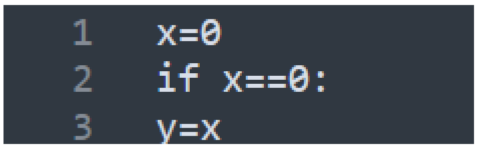
I collect previous **tabCounter** as **tempTabCounter** and my present tab counter is **tabCounter**. When I get into **“if, elif and else”** statmenent, I increment my **tabCounter tempTabCounter +1.** It is because I can just put **1 tab** when I get into **if statements**. So, I check in the **assignment** part if **tempTabCounter < tabCounter,** it means there must be a tab inconsistency. In this example **tempTabCounter** is 1 and **tabCounter is** 2. So the program will print, **tab inconsistency in line xx.**

2) InvalidOutput2.txt



```
user@DESKTOP-U5L2IAV:/mnt/c/Users/Mehmet/Desktop/termProject/diff$ diff -s invalidOutput2.txt invalid2outputStudent.txt
Files invalidOutput2.txt and invalid2outputStudent.txt are identical
```

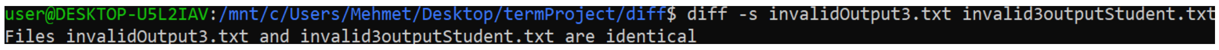
I collect all the if else and else if statement in a struct and I collect its tab counter, it is closed or not and it is filled or not.



```
1 x=0
2 if x==0:
3 y=x
```

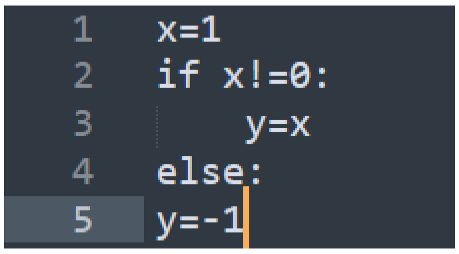
For this example, I collect first if and its **“tab counter”** as **“1”**. So, in the **3rd line**, I check if above if is not **closed and filled** and my present **tab counter** and “if tab counter” are same. It means there is nothing in if statement. When I catch this condition, I print **error in line xx: at least one line should be inside if/elif/else block.**

3) InvalidOutput3.txt



```
user@DESKTOP-U5L2IAV:/mnt/c/Users/Mehmet/Desktop/termProject/diff$ diff -s invalidOutput3.txt invalid3outputStudent.txt
Files invalidOutput3.txt and invalid3outputStudent.txt are identical
```

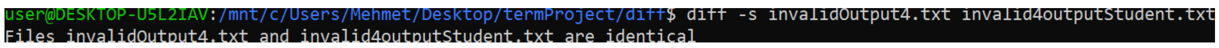
It is same as with the example invalidOutput2.txt



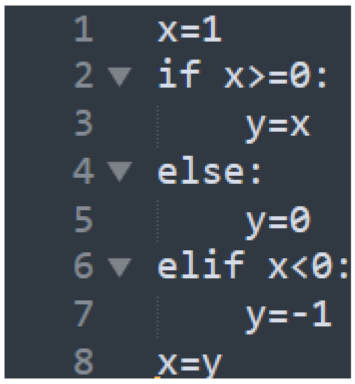
```
1  x=1
2  if x!=0:
3      y=x
4  else:
5  y=-1
```

For this example, I collect **else** and its **"tab counter"** as **"1"**. So, in the **5th line**, I check if above if is not **closed and filled** and my present **tab counter** and "**else** tab counter" are same. It means there is nothing in **else** statement. When I catch this condition, I print **error in line xx: at least one line should be inside if/elif/else block.**

4) InvalidOutput4.txt



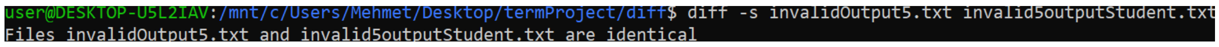
```
user@DESKTOP-U5L2IAV:/mnt/c/Users/Mehmet/Desktop/termProject/diff$ diff -s invalidOutput4.txt invalid4outputStudent.txt
Files invalidOutput4.txt and invalid4outputStudent.txt are identical
```



```
x=1
if x>=0:
    y=x
else:
    y=0
elif x<0:
    y=-1
x=y
```

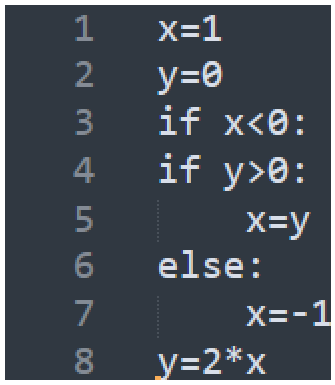
In this example, I'm checking all the **"If elif and else"** statement. Then I check whether the present **tabCounter** and **"If elif and else"**'s **tabCounter** is same. When they're same and it is **else** statement, it means **elif** is coming after **else**. When I catch this condition, The program will print **elif after else in line xx.**

5) InvalidOutput5.txt



```
user@DESKTOP-U5L21AV:/mnt/c/Users/Mehmet/Desktop/termProject/diff$ diff -s invalidOutput5.txt invalid5outputStudent.txt
Files invalidOutput5.txt and invalid5outputStudent.txt are identical
```

It is same as with the example **invalidOutput2.txt** and **invalidOutput3.txt.**

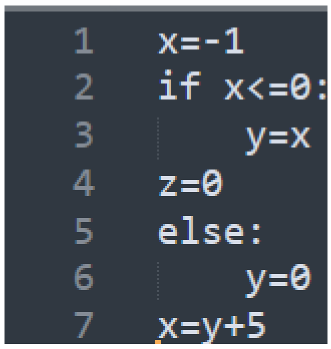


```
1  x=1
2  y=0
3  if x<0:
4  if y>0:
5      x=y
6  else:
7      x=-1
8  y=2*x
```

As you see in the 3rd line I collect if statement and it's situation is open. Then in 4th line there is another if statement but their **tabCounter** are same. Thus, The program will print **error in line xx: at least one line should be inside if/elif/else block.**

6) InvalidOutput6.txt

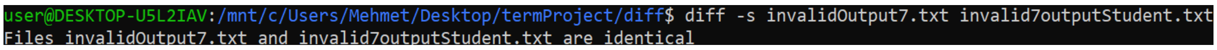
I collect all the if else and else if statement in a struct and I collect its tab counter, it is closed or not and it is filled or not.



```
1  x=-1
2  if x<=0:
3      y=x
4  z=0
5  else:
6      y=0
7  x=y+5
```

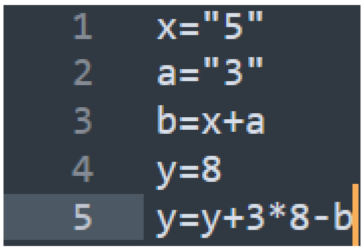
For this example, In the **5th** line, There is an else statement but its not a part of any if statement because in the **4th** line there is an assignment and it closes the if statement its above. That's why the program will print **else without if in line.**

7) InvalidOutput7.txt



```
user@DESKTOP-U5L2IAV:/mnt/c/Users/Mehmet/Desktop/termProject/diff$ diff -s invalidOutput7.txt invalid7outputStudent.txt
Files invalidOutput7.txt and invalid7outputStudent.txt are identical
```

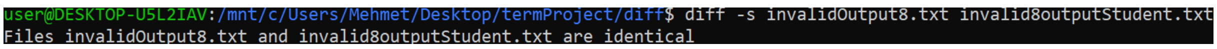
I collect all the identifiers and their types in a vector.



```
1   x="5"
2   a="3"
3   b=x+a
4   y=8
5   y=y+3*8-b
```

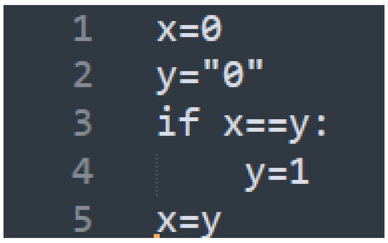
For this example, in the 3rd line b will be collected as **string** because “**x**” and **“a”** is **string**. Thus, in the **5th** line the program catches an error because **“b”** is **string**, but the other variables are integer. So, it will print **type mismatch in line xx.**

8) InvalidOutput8.txt



```
user@DESKTOP-U5L2IAV:/mnt/c/Users/Mehmet/Desktop/termProject/diff$ diff -s invalidOutput8.txt invalid8outputStudent.txt
Files invalidOutput8.txt and invalid8outputStudent.txt are identical
```

I collect the types of variables in statements in a temp vector. So all I need to do is checking in the temp vector, if there is a different variables type such as **“flt” and “str” , “int” and “str”.** the program will print error. But when it is **“flt” and “int”** the variable type will turn into **“flt”** and the program will **continue**.

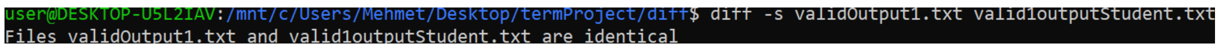


```
1   x=0
2   y="0"
3   if x==y:
4       y=1
5   x=y
```

For this example, x is collected as **“int”** and y as **“str”.** So, in **3rd** line if statement will try to compare **“int”** and **“str”.** So, the program will print an **error comparison type mismatch in line xx.**

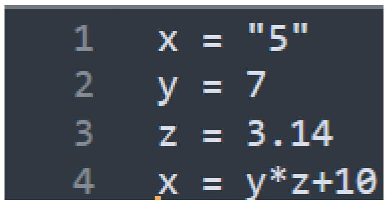
For all “valid Identical” part:

1) **ValidOutput1.txt**



```
user@DESKTOP-U5L2IAV:/mnt/c/Users/Mehmet/Desktop/termProject/diff$ diff -s validOutput1.txt valid1outputStudent.txt
Files validOutput1.txt and valid1outputStudent.txt are identical
```

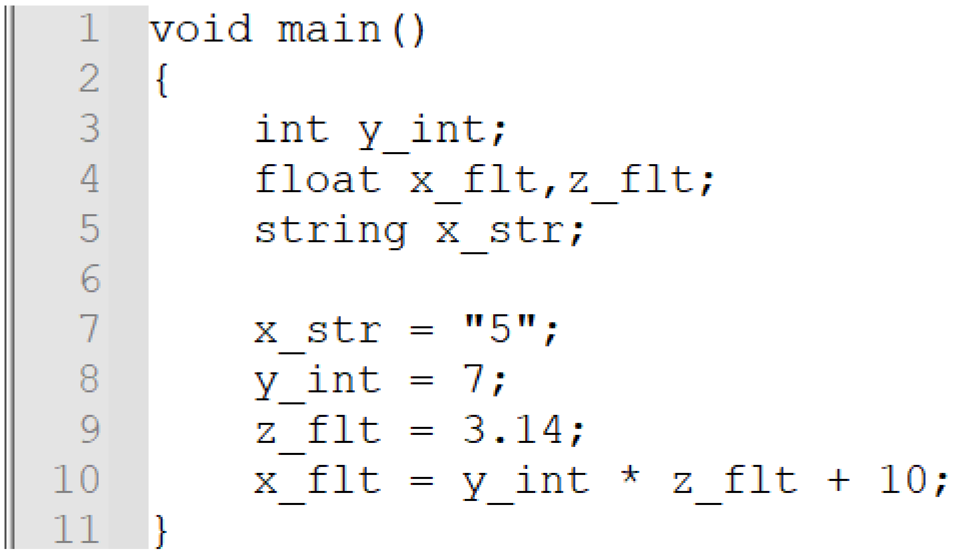
Input:



```
x = "5"
y = 7
z = 3.14
x = y*z+10
```

For all the printing part I collect every line in a vector. Then I print in the **“statement”** part. First I print void main() part. I sort **varsLast** vector. Then I print variables part that I get them from **vector<variables> varsLast** that they’re **collected** in **assignment** part. Then I print whole lines in **vector<string> lines**. I collect all the lines where they’re collected in **assignment and if elif else statements.**

My output:



```
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;
}
```

In this example, I could collect the variables name and variables type in a vector. Normally, all the variables don’t have _str, _int, _flt part. I collect their type as **string** so I just **push_back** into the **lines** vector as **variablesName_variablesType.**

All in all, I explained why I didn’t get same output for 2 valid inputs. Then I explained why I get same output for all the other.