CSE351 2023-Fall Project Report

All inputs are converted c++ successfully. And also whole of the invalid inputs are successfully analysed.

In order to check inconsistency in codes, in lex codes I count the number of tab. When the yacc found a if, else if or else text, I increment the usable tab number 1. And if number of TAB used in code is much than usable tab then there should be tab inconsistency.

Below code return TAB count in given code

```
(\t|[ ]{4})+ { //Pythonda bir tab 4 boşluğa eşit olduğundan 4 kullanıldı.
    tabcount =0;
    for(int i =0;i<=strlen(yytext);i++)
        if(yytext[i]=='\t')
            tabcount++;
    yylval.tab = tabcount;
    return TAB; //Koddaki tab sayısını döndürüyor
}
```

I prioritized the handling of "if-else" statements using a boolean variable. When an "if" statement is encountered and opens, I set this variable to true. If subsequent lines of code do not match the expected indentation level determined by the usable tab variable, the program generates an error message.

In invalid2.txt, invalid3.txt, invalid5.txt we can see if else elif blocks have lack of content problem.

```
statement
{
if (counter_if > 0) {
    string temp;
    int i = 0;
    while (i < counter_if - 1) {
        temp += "\t";
        ++i;
    }

    temp = temp + "}\n" + string($1.name);
    $1.name = strdup(temp.c_str());

    counter_if = 0;
}


if(ifopened){
    cout << "error in line "<<linenumber<<": at least one line should be inside if/elif/else block "<<
    return 0;
}
if($1.type)
{
    counter_if++;
    ifopened = true;
    string combined1 = string($1.name) + "\n{";
    $1.name = strdup(combined1.c_str());
}
else{
   ifopened=false;
    }

string temp = string($1.name);
$$ = strdup(temp.c_str());
}
```

And also I checked the order of if elseif else blocks. Using else and elseif without using if is not allowed in C so YACC file will give necessery erorrs in those statements. We can test those statements with invalid4.txt, invalid6.txt. I used int variable to check if “if loop is started or not” when else comes loopstarterchech variable will be 0 in order to show if loop is finished and also I store the count of if with counter_if variable, with this variable I can check the order of if else.

```
ifelse:
    IF condution COLUMN
    {
        loopstartcheckher = 1;
        string combined = "if" + string($2);
        $$ = strdup(combined.c_str());

    }
    |
    ELIF condution COLUMN
    {
        if(counter_if ==0 ){
            cout << "else without if in line "<< linenumber << endl;
            return 0 ;
        }
        else if(loopstartcheckher!=1)
        {
            cout << "elif after else in line " << linenumber<<endl;
            return 0;
        }
        string temp = "else if" + string($2);
        $$ = strdup(temp.c_str());
    }
    |
    ELSE COLUMN
    {
        loopstartcheckher = 0;
        if(counter_if ==0){
            cout << "else without if in line "<< linenumber << endl;
            return 0 ;
        }
        string temp = "else";
        $$ = strdup(temp.c_str());
    }
    ;
```

To handle variable problems I created a struct named var in order to assign variables names and types. And also I used 2 different maps, I stored all variables in a list according to their types, and the other map I stored type of variables. For instance “int b = 5”, in first map I stored b variable with name in a integ er list, and in second map I stored the type of the b variable in order to check iftype mismatching error exist in condition. And we can test variable issues with invalid7.txt and invalid6.txt

```
expr:VARIABLE
    {
        int ty = current_var[string($1)]; //Storing variable actual type.
        $$.type = ty;
        string combined = string($1)+"_"+int_typestr(ty);
        $$.name = strdup(combined.c_str());
    }
    |
    type
    {
        $$.type=$1.type;
        string combined =string($1.name);
        $$.name = strdup(combined.c_str());
    }
    |
    expr OPERATOR expr
    {

        string combined = string($$.name)+" "+$2+" " +string($3.name);
        $$.name = strdup(combined.c_str());

        if($1.type==$3.type)
            $$.type=$1.type;
        else if( $1.type==1 && $3.type==0 )
            $$.type=1;
        else if( $1.type==0 && $3.type==1 )
            $$.type=1;
        else{
            cout << "type mismatch in line "<< linenumber <<endl;
            return 0;
        }

    }
    ;
```

```
344   type:
345       INT
346       {
347           string temp = string($1);
348           $$.name = strdup(temp.c_str());
349           $$.type = 0;
350       }
351       |
352       FLT
353       {
354           string temp = string($1);
355           $$.name = strdup(temp.c_str());
356           $$.type = 1;
357       }
358       |
359       STR
360       {
361           string temp = string($1);
362           $$.name = strdup(temp.c_str());
363           $$.type = 2;
364       }
365       ;
366
367
368
369   %%
370
371   string int_typestr(int a){
372
373       if(a==0)
374           return "int";
375       if(a==1)
376           return "flt";
377       else
378           return "str";
379
380
381   }
```