**code.1**

```
443
444 %x HEADER
445 %x COMMENT
446 %x MAIN
447 %x VAR_SEC
448 %x VAR_CALC_SEC
449 %x IF_SEC
450 %x IGNORE_SEC
451 %x DISCARD_SEC
452 %x LOOP_SEC
453 %x OUT_SEC
454
455 DQ \"
456 NUMBER ([0-9]+(".")[0-9]+)|([0-9]+)|("."[0-9]+)
457 VARIABLE [a-zA-Z][a-zA-Z0-9]*
458 VAR_NUM ({NUMBER}|{VARIABLE})
459 COND_OP ("lt"|"gt"|"eq"|"neq"|"le"|"ge")
460 ARITH_OP ("add"|"sub"|"mul"|"div"|"dif"|"rem")
461
462 HEADER_START "import "
463 HEADER_END [a-zA-Z.]+";"
464
465 OUT_START "println("[ ]*
466 OUT_BODY_CONST [ ]*{DQ}[^\"]*{DQ}(,)?[ ]*
467 OUT_BODY_VAR [ ]*{VAR_NUM}[ ]*(,)?
468 OUT_END [ ]*")"[ ]*;[ ]*
469
470 MAIN_START "static void entryPoint"[ ]*"("[ ]*")"[ ]*"{"[ ]*
471 MAIN_END "}"
472
473 SINGLE_LINE_COMMENT ("//").*(\n)?
474
475 VARIABLE_ONLY [ ]*{VARIABLE}*[ ]*(,)
476 VARIABLE_VALUE [ ]*{VARIABLE}[ ]*("=")[ ]*{NUMBER}(,)
477 VARIABLE_VALUE_ASSIGN_CONST [ ]*{VARIABLE}[ ]*("=")[ ]*{NUMBER};
478 VARIABLE_VALUE_ASSIGN_VAR [ ]*{VARIABLE}[ ]*("=")[ ]*{VARIABLE};
479 VARIABLE_VALUE_ASSIGN_CALC [ ]*{VARIABLE}[ ]*("=")[ ]*{VAR_NUM}[ ]*{ARITH_OP}[ ]*{VAR_NUM}[ ]
    *,
480 VARIABLE_VALUE_ASSIGN_CALC_LAST [ ]*{VARIABLE}[ ]*("=")[ ]*{VAR_NUM}[ ]*{ARITH_OP}[ ]*
    {VAR_NUM}[ ]*;
481
482 DISCARD_START "discard "
483
484 VARIABLE_ONLY_LAST [ ]*[a-zA-Z][a-zA-Z0-9]*[ ]*(;)
485 VARIABLE_VALUE_LAST [ ]*[a-zA-Z][a-zA-Z0-9]*[ ]*("=")[ ]*{NUMBER}(;)
486
487 VAR_SPACE [ ]*
488
489 DATA_TYPE ("int"|"float"|"double")[ ]
490
491 IF "justInCase"[ ]*
492 IF_COND ("(")[ ]*{VAR_NUM}[ ]*{COND_OP}[ ]*{VAR_NUM}[ ]*(")")[ ]*
493 IF_BODY_START [ ]*"{"
494 IF_BODY_END [ ]*"}"
495 IF_SPACE [ ]*
496
497 IGNORE_LEFT_BRACE "{"
498 IGNORE_RIGHT_BRACE "}"
499
500 LOOP_START "till"
501 LOOP_START_BRACE "{"
502 LOOP_COND {IF_COND}
503 LOOP_OTHERS [^{}\n]*\n
504 LOOP_END {IF_BODY_END}
505
506 NEW_LINE_AND_TAB [\n\t]*
507
508 %%
509 {HEADER_START} { BEGIN(pushState(HEADER)); }
510
511 <HEADER>{HEADER_END} { inc(0); runHeader(1); BEGIN( popState() ); }
512
513 <HEADER>. { runHeader(0); BEGIN(popState()); }
514
515 {SINGLE_LINE_COMMENT} { inc(1); initSingleComment(); }
516
517 "/*" { initMultiComment(); BEGIN( pushState(COMMENT) ); }
518
519 <COMMENT>"/*" { appendTextToBuffer("/*"); }
520 <COMMENT>"*/" {
521     appendTextToBuffer("*/"); checkForEnd();
522     if(comment_depth == 0){
523         inc(2);
524         BEGIN( popState() );
525     }
526 }
527
528 <COMMENT>. { appendToBuffer(yytext[0]); }
529 <COMMENT>\n { appendTextToBuffer("\n"); }
530
531 <VAR_SEC>{VARIABLE_ONLY} { declareVariable(yytext,0); }
532 <VAR_SEC>{VARIABLE_ONLY_LAST} {
533     declareVariable(yytext,0);
534     BEGIN( popState() );
535 }
536 <VAR_SEC>{VARIABLE_VALUE_ASSIGN_CALC} {
537     //printf("\nassign from calc: %s\n",yytext);
538     continueVariableCalc(yytext,true);
539 }
540 <VAR_SEC>{VARIABLE_VALUE_ASSIGN_CALC_LAST} {
541     //printf("\nassign from calc: %s\n",yytext);
542     inc(6);
543     continueVariableCalc(yytext,true);
544     BEGIN( popState() );
545 }
546 <VAR_SEC>{VARIABLE_VALUE} { processVariable(yytext); }
547 <VAR_SEC>{VARIABLE_VALUE_LAST} {
548     //printf("\nvariable value last-%s-\n",yytext);
549     processVariable(yytext);
550     BEGIN( popState() );
```

```
}
<VAR_SEC>{VAR_SPACE} {}
<VAR_SEC>. {
    printf("\nInvalid %s\n",yytext);
    stopProgram("Invalid character found");
}
<VAR_SEC>"/*" {
    initMultiComment();
    BEGIN( pushState(COMMENT) );
}


<MAIN>{IF} { inc(7); BEGIN( pushState(IF_SEC) ); }

<IF_SEC>{IF_COND} { inc(9); processIfBody(yytext); }
<IF_SEC>{IF_BODY_START} {
    blockBalancer++;
    popState(); // popping if section and taking to main section arki
    if(isLastIfValid){
        BEGIN(pushState(MAIN));
    }
    else{
        bracketCounter=1;
        BEGIN( pushState(IGNORE_SEC) );
    }
}

<IF_SEC>. {}
<IGNORE_SEC>{IGNORE_LEFT_BRACE} { blockBalancer++; bracketCounter++; }
<IGNORE_SEC>{IGNORE_RIGHT_BRACE} {
    blockBalancer--;
    bracketCounter--;
    if(bracketCounter == 0){
        BEGIN( popState() );
    }
}
<IGNORE_SEC>[^}] {}

{MAIN_START} {
    blockBalancer++;
    //printf("main start %s\n",yytext);
    initMain();
    BEGIN( pushState(MAIN) );
}
<MAIN>{DATA_TYPE} {
    //printf("main data type %s\n",yytext);
    initVarSec(yytext);
    BEGIN( pushState(VAR_SEC) );
}
<MAIN>{VARIABLE_VALUE_ASSIGN_CONST} {
    //printf("main assign %s\n",yytext);
    updateValue(yytext);
}
<MAIN>{VARIABLE_VALUE_ASSIGN_VAR} {
    //printf("\nassign from var: %s\n",yytext);
    updateValue(yytext);
}

<MAIN>{VARIABLE_VALUE_ASSIGN_CALC_LAST} {
    inc(6);
    //printf("\nassign from calc: %s\n",yytext);
    continueVariableCalc(yytext,false);
}

<MAIN>{OUT_START} { initOutBuffer(); BEGIN( pushState(OUT_SEC) ); }

<OUT_SEC>{OUT_BODY_CONST} { continueOutBuffer(yytext); }
<OUT_SEC>{OUT_BODY_VAR} { continueOutBuffer(yytext); }
<OUT_SEC>{OUT_END} { inc(4); BEGIN( popState() ); printAndClearOutBuffer(); }

<MAIN>{MAIN_END} {
    blockBalancer--;
    //stopMain();
    //int val = popState();
    //printf(" main end popped %s %d %d\n",yytext,val,MAIN);
    BEGIN( popState() );
}
<MAIN>{SINGLE_LINE_COMMENT} {
    inc(1);
    //printf("main comment %s\n",yytext);
    initSingleComment();
}
<MAIN>"/*" {
    //printf("main multi %s\n",yytext);
    initMultiComment();
    BEGIN( pushState(COMMENT) );
}
<MAIN>{NEW_LINE_AND_TAB}* { }

<MAIN>{DISCARD_START} {
    //printf("Discard section\n");
    inc(5);
    BEGIN( pushState(DISCARD_SEC) );
}

<MAIN>{LOOP_START} {
    inc(8);
    blockBalancer++;
    printf("\nloop start %s\n",yytext);
    BEGIN( pushState(LOOP_SEC) );
}
<MAIN>[^ ] {
    //printf("-%s-\n",yytext);
    stopProgram("Invalid character found\n");
}

<LOOP_SEC>{LOOP_COND} { inc(9); printf("\nloop cond: %s\n",yytext); }
<LOOP_SEC>{LOOP_START_BRACE} { printf("\nloop start {: %s\n",yytext); }
<LOOP_SEC>{LOOP_END} {
    blockBalancer--;
    printf("\nloop end: %s\n",yytext);
    BEGIN( popState() );
```

```
}
<LOOP_SEC>{NEW_LINE_AND_TAB} {}
<LOOP_SEC>{LOOP_OTHERS} { printf("\nloop others: %s\n",yytext); }

<DISCARD_SEC>{VARIABLE_ONLY} {
    //printf("\nFrom discard only:-%s-\n",yytext);
    discardVariable(yytext);
}
<DISCARD_SEC>{VARIABLE_ONLY_LAST} {
    //printf("\nFrom discard last:-%s-\n",yytext);
    discardVariable(yytext);
    int state = popState();
    //printf("\nState %d %d\n",MAIN,state);
    BEGIN( state );
}

<DISCARD_SEC>. { stopProgram("\nInvalid character while discarding\n"); }
. {
    printf("invalid -%s-\n",yytext);
    stopProgram("Invalid char outside block");
}
{NEW_LINE_AND_TAB}* { }

%%

int yywrap(){
    return 1;
}


int main(){

    yyin = fopen("input.txt","r");
    yylex();

    if(blockBalancer != 0){
        stopProgram("\nBlock not completed");
    }

    tempCounter[3] = getTotalVar();
    printAll();

    if( strlen(error) == 0 ){
        printf("Total comment: %d\n",commentCounter);
        printf("Program compiled successfully\n");
    }
    else{
        printf("Error: %s\n",error);
    }



    printf("\n----------------- Summary ------------------\n\n");
    printf("%-20s %d\n","Header:",tempCounter[0]);
    printf("%-20s %d\n","Single line comment:",tempCounter[1]);
    printf("%-20s %d\n","Multi line comment:",tempCounter[2]);
    printf("%-20s %d\n","Variable:",tempCounter[3]);
    printf("%-20s %d\n","println:",tempCounter[4]);
    printf("%-20s %d\n","discard:",tempCounter[5]);
    printf("%-20s %d\n","val_calculation:",tempCounter[6]);
    printf("%-20s %d\n","justInCase:",tempCounter[7]);
    printf("%-20s %d\n","till:",tempCounter[8]);
    printf("%-20s %d\n","condition:",tempCounter[9]);

}
```