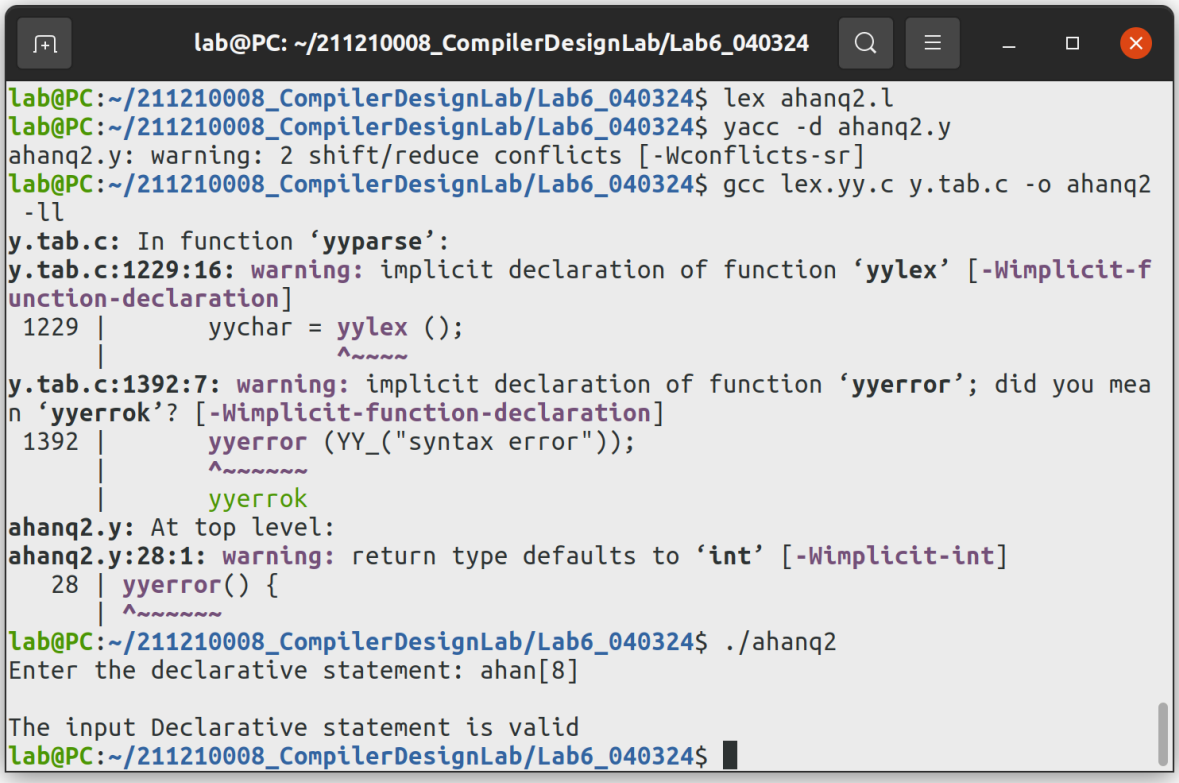
1. Design a grammar for a declarative statement for C program. Further, write a Yacc program to check if the entered statement is a valid declarative statement according to the grammar generated.

Yacc file

```
%{
#include <stdio.h>
#include <stdlib.h>
int flag = 0;
%}

%token ID KEY COLON COMMA NUM

%%
stmt: list{printf("\nThe input Declarative statement is valid\n");};
list : KEY list
        | list ',' list
        | list ',' ',' {printf("Consecutive commas invalid.\n");exit(0);}
        | ID '[' NUM ']'
        | ID '[' NUM '.' ']' {printf("Float number canNOT be the size of an array.\n");exit(0);}
        | ID '[' ID ']' {printf("Size should be integer.\n");exit(0);}
        | ID '[' ID  {printf("Missing closing parenthesis.\n");exit(0);}
        | ID '['  {printf("Missing size of array.\n");exit(0);}
        | ID
        ;
%%

int main(){
  printf("Enter the declarative statement: ");
  yyparse();
}

yyerror() {
   printf("Invalid declarative statement.\n");
   exit(1);
}
```

Lex file

```
%{
#include "y.tab.h"
#include<stdio.h>
#include <stdlib.h>
%}

%%
"int" {return KEY;}
"float" {return KEY;}
"char" {return KEY;}
"double" {return KEY;}
[a-zA-Z][a-zA-Z0-9]* {return ID;}
[0-9]+ {return NUM;}
[\t] ;
[;] {return COLON;}
\n {return 0;}
. {return yytext[0];}
%%

int yywrap(){
  return 1;
}
```



2. Design a grammar for a relational expression of C language. Further, write a Yacc program to check if the entered statement is a valid relational expression according to the grammar generated.

Yacc file

```
%{
  /* Definition section */
  #include<stdio.h>
  #include<stdlib.h>
%}

%token A B NL

/* Rule Section */
%%
stmt: S NL  { printf("valid string\n");
        exit(0); }
;
S: A S B |
;
%%

int yyerror(char *msg)
```

```
 {
  printf("invalid string\n");
  exit(0);
 }

//driver code
main()
 {
  printf("enter the string\n");
  yyparse();
 }
```

Lex file

```
%{
  #include "y.tab.h"
%}

/* Rule Section */
%%
[aA] {return A;}
[bB] {return B;}
\n {return NL;}
```

. {return yytext[0];}

<<EOF>> {return 0;}

%%

int yywrap()

{

return 1;

}

```
lab@PC:~/211210008_CompilerDesignLab/Lab6_040324$ lex ahanq3.l
lab@PC:~/211210008_CompilerDesignLab/Lab6_040324$ yacc -d ahanq3.y
lab@PC:~/211210008_CompilerDesignLab/Lab6_040324$ gcc lex.yy.c y.tab.c -o ahanq3 -ll
y.tab.c: In function ‘yyparse’:
y.tab.c:1215:16: warning: implicit declaration of function ‘yylex’ [-Wimplicit-function-declaration]
 1215 |       yychar = yylex ();
      |                ^~~~~
y.tab.c:1349:7: warning: implicit declaration of function ‘yyerror’; did you mean ‘yyerrok’? [-Wimplicit-function-declaration]
 1349 |       yyerror (YY_("syntax error"));
      |       ^~~~~~~
      |       yyerrok
ahanq3.y: At top level:
ahanq3.y:25:1: warning: return type defaults to ‘int’ [-Wimplicit-int]
   25 | main()
      | ^~~~
lab@PC:~/211210008_CompilerDesignLab/Lab6_040324$ ./ahanq3
enter the string
aaaaabbbbb
valid string
lab@PC:~/211210008_CompilerDesignLab/Lab6_040324$ ./ahanq3
enter the string
abbbb
invalid string
lab@PC:~/211210008_CompilerDesignLab/Lab6_040324$
```