## 1. Convert the given switch case statement to else if statement.

**CODE :-**

```
%option noyywrap
%{
        #include<stdio.h>
        int i, first=0;
        char arr[50];
%}
%%
"switch("[a-zA-Z_][a-zA-Z0-9_]*")" {
        for(i=7; i<yyleng-1; i++)
                arr[i-7]=yytext[i];
        arr[i]='\0';
}
"case ""\'"?[a-zA-Z_][a-zA-Z0-9_]*"\'"?":" {
        if(first){fprintf(yyout,"else ");}
        fprintf(yyout,"if(%s == ",arr);
        for(i=5; i<yyleng-1; i++)
                fprintf(yyout,"%c",yytext[i]);
        fprintf(yyout,"){\n");
        first++;
}
[ \t\n]*([a-zA-Z0-9]+";"[ \t\n]*)+"break;" {
        for(i=0; i<yyleng-8; i++)
                fprintf(yyout,"%c",yytext[i]);
        fprintf(yyout,"}\n");
}
"default:"[ \t\n]*([a-zA-Z0-9]+";"[ \t\n]*)+"}" {
        fprintf(yyout,"else{");
        for(i=8; i<yyleng-1; i++)
                fprintf(yyout,"%c",yytext[i]);
        fprintf(yyout,"}");
}
. ;
%%

void main()
{
        extern FILE *yyin, *yyout;
        yyin=fopen("input.txt", "r");
        yyout=fopen("output.txt", "w");
        yylex();
}

```

**OUTPUT :-**

```
C:\WINDOWS\system32\cmd.exe

D:\STUDIES\SEM 5\CD\LAB\CODE\LAB 6>lex p1.l

D:\STUDIES\SEM 5\CD\LAB\CODE\LAB 6>gcc lex.yy.c

D:\STUDIES\SEM 5\CD\LAB\CODE\LAB 6>a.exe

D:\STUDIES\SEM 5\CD\LAB\CODE\LAB 6>type input.txt
switch (expression)

{
        case value1:
                statement1;
                break;
        case value2:
                statement2;
                break;
        default:
                statementDefault;
}
D:\STUDIES\SEM 5\CD\LAB\CODE\LAB 6>type output.txt



if(expression  == value1){

                statement1;
}

else if(expression  == value2){

                statement2;
}

else{
                statementDefault;
}
D:\STUDIES\SEM 5\CD\LAB\CODE\LAB 6>
```

## 2. Write a program to convert the given function to macro.

**CODE :-**

```
%option noyywrap
%{
        #include<stdio.h>
        #include<string.h>
        int i;
%}
datatype "int"|"char"|"float"|"double"
%%
" "[a-zA-Z_][a-zA-Z0-9_]*"(" {
        fprintf(yyout,"#define ");
        for(i=1;i<yyleng;i++)
                fprintf(yyout,"%c",toupper(yytext[i]));
}
(datatype)?" "[a-zA-Z_][a-zA-Z0-9_]*(")"|",") {
        for(i=yyleng-2; yytext[i]!=' ' && i>=0; i--)
                fprintf(yyout,"%c",yytext[i]);
        if(yytext[yyleng-1]==',')
                fprintf(yyout,",",yytext);
}
[\n] ;
"return "[a-zA-Z0-9*+-/\)\(]+";" {
        fprintf(yyout,") ",yytext);
        for(i=7; i<yyleng-1; i++)
                fprintf(yyout,"%c",yytext[i]);
}
. ;
%%
int main()
{
        extern FILE *yyin, *yyout;
        yyin=fopen("input2.txt", "r");
        yyout=fopen("output2.txt", "w");
        yylex();
        return 0;
}
```

**OUTPUT :-**

```
D:\STUDIES\SEM 5\CD\LAB\CODE\LAB 6>lex p2.l

D:\STUDIES\SEM 5\CD\LAB\CODE\LAB 6>a.exe

D:\STUDIES\SEM 5\CD\LAB\CODE\LAB 6>type input2.txt
int area_square(int s)
{
        return s*s;
}
D:\STUDIES\SEM 5\CD\LAB\CODE\LAB 6>type output2.txt
#define AREA_SQUARE(s) s*s
D:\STUDIES\SEM 5\CD\LAB\CODE\LAB 6>
```

## 3. To write a program to convert the given infix expression to postfix expression(with parenthesis).

**CODE :-**

```
%{
 #include<stdio.h>
 #include<string.h>
 char stack[100];
 int tos=0;
 void push(char);
 void pop();
 char top();
 int priority(char c);
%}
%%
[a-zA-Z0-9] { printf("%s",yytext);}
[+\-*/^()] {
 char op=yytext[0];
 if(tos==0||op=='(')
 push(op);
 else if(op==')')
 {
 while(top()!='(')
 {
 printf("%c",top());
 pop();
 }
 pop();
 }
 else if(priority(op) >= priority(top()))
 push(op);
 else if(priority(op) <= priority(top()))
 {
 while(priority(op) <= priority(top()))
 {
 printf("%c",top());
 pop();
 }
 push(op);
 }
}
"\n" {
 int i;
 for(i=tos-1;i>=0;i--)
 if(stack[i]!='(' && stack[i]!=')')
 printf("%c",stack[i]);
 tos=0;
 printf("\n");
 return 0;
}
%%
int yywrap() {}
void main()
{
 printf("Enter infix expression: ");
 yylex();
}
```

```
54  void pop()
55  {
56   tos--;
57  }
58  char top()
59  {
60   return stack[tos-1];
61  }
62  void push(char c)
63  {
64   stack[tos]=c;
65   tos++;
66  }
67  int priority(char c)
68  {
69   switch(c)
70   {
71   case '(':return 0;
72   case ')':return 0;
73   case '+':return 1;
74   case '-':return 1;
75   case '*':return 2;
76   case '/':return 2;
77   case '^':return 3;
78   }
79  }
```

**OUTPUT :-**

```
D:\STUDIES\SEM 5\CD\LAB\CODE\LAB 6>a.exe

D:\STUDIES\SEM 5\CD\LAB\CODE\LAB 6>lex p3.l

D:\STUDIES\SEM 5\CD\LAB\CODE\LAB 6>gcc lex.yy.c

D:\STUDIES\SEM 5\CD\LAB\CODE\LAB 6>a.exe
Enter infix expression: a+b
ab+

D:\STUDIES\SEM 5\CD\LAB\CODE\LAB 6>a.exe
Enter infix expression: a-b*c
abc*-

D:\STUDIES\SEM 5\CD\LAB\CODE\LAB 6>a.exe
Enter infix expression: (a+b)*c
ab+c*

D:\STUDIES\SEM 5\CD\LAB\CODE\LAB 6>a.exe
Enter infix expression: (a+b)*(c+d)
ab+cd+*

D:\STUDIES\SEM 5\CD\LAB\CODE\LAB 6>a.exe
Enter infix expression: a-b*c/d
abcd/*-

D:\STUDIES\SEM 5\CD\LAB\CODE\LAB 6>a.exe
Enter infix expression: a+b-c*d/e*f
abcdef*/*-+
```