INTRODUCTION:

Solving electrical circuits is an important step in the field of Electronics and Communication Engineering. The calculation of resistance between two points is a part of solving these circuits. The calculation of equivalent resistance between 2 points is used to find the voltages and currents through the entire circuit. Here I used the **STACK** data structure to find the equivalent resistance between two points.

[Ohm meter is a device which measures the resistance between 2 points in the circuit]

ALGORITHM:

Step 1: Form an expression between any two points from the circuit.

Step 2: Enter the expression as input to the program.

Step 3: The given expression contains S for series and P for parallel. So we need to replace those with equivalent symbols.

Step 4: Convert the given expression to postfix.

Step 5: Evaluate the postfix expression. This gives the equivalent resistance between the 2 points.

Step 6: End.

MAIN FORMULAE USED:

If 2 resistances are in series their equivalent resistance is

$$R_{eq} = R1+R2$$

If 2 resistances are in parallel their equivalent resistance is

$$R_{eq} = 1/((1/R1)+(1/R2))$$

DETAILS ON THE TOPIC:

- An ohm meter is a device which measures the resistance between two points in the given circuit.
- Here I designed an ohm meter (digitally) which measures the resistance between any two points on the given circuit.

ADVANTAGES OF THIS PROGRAM:

- This program can be used as a basic part for designing simulation software like multisim etc.
- This program can be used as a basic for android app development which is used to find the equivalent resistance.

DISADVANTAGES OF THIS PROGRAM:

- This program can be used only for resistance calculation (not other passive components like capacitor, inductor).
- This program needs the resistance equation for calculating equivalent resistance.

ERROR AVOIDED:

- The most common error while entering the equation is the missing parenthesis. This program avoids that error by checking whether the equation is balanced or not.

CONDITIONS FOR ENTERING EQUATION:

- Usage of '(' &')' type of brackets are only allowed(usage of other type of parenthesis should be avoided)
- For series connection use 'S' and for parallel connection use 'P'.
  e.g 12S(12P12) means 12 ohm resistor is in series with a parallel combination of 2 12 ohm resistors.

```c
#include<stdio.h>
#include<ctype.h>
#include<stdlib.h>
#define max 100
char stack1[max];
int tos1 = -1;
char infix[max],postfix[max],stack[max];
int tos = -1;
float member[max];
void push1(char element){

    if (tos1 == max-1){
        printf("Stack overflow. Stack can hold a maximum of 20 characters \n");
    }
    else{
        tos1++;
        stack1[tos1] = element;
    }
}

char pop1(){
    if (tos1 == -1){
        printf("Stack underflow\n");
    }
    else{
        return stack1[tos1--];
    }
}

void pushfl(float element)
{
    if (tos == max-1){
        printf("Stack Overflow\n");
    }
    else{
        member[++tos]=element;
    }

}
float popfl(){

return(member[tos--]);


}
int prec(char element){

    if(element ==  '('){
        return 1;
    }
```

```
51
52      if(element ==  '+' ){
53          return 2;
54      }
55
56      if(element == '/'){
57          return 3;
58      }
59
60  }
61  void push(char element)
62  {
63      if (tos == max-1){
64          printf("Stack Overflow\n");
65      }
66      else{
67          stack[++tos]=element;
68      }
69
70  }
71  char pop(){
72      if (tos == -1){
73          printf("Stack Underflow\n");
74      }
75      else{
76          return(stack[tos--]);
77      }
78
79  }
80
81  int balanced(){
82      int i,flag = 1;
83
84      for (i = 0;i < strlen(infix);++i){
85          if (infix[i] == '(' || infix[i] == '{' || infix[i] == '['){
86              push1(infix[i]);
87          }
88          if (infix[i] == ')' || infix[i] == '}' || infix[i] == ']'){
89              if (infix[i] == ')'){
90                  if (pop1() != '('){
91                      flag = 0;
92                  }
93              }
94              if (infix[i] == '}'){
95                  if (pop1() != '{'){
96                      flag = 0;
97                  }
98              }
99              if (infix[i] == ']'){
100                 if (pop1() != '['){
```

```
                flag = 0;
            }
        }
    }


}

    if (flag == 1 && tos1 == -1){
        return 1;
    }
    else{
        return 0;
    }
}

void evalpost(){
    int i=0,j=0;
    char sub[max];
    while(postfix[i] != '\0'){
        if (postfix[i] == '+' || postfix[i] == '/'){
            if (postfix[i] == '+'){
                pushfl(popfl()+popfl());

            }
            if (postfix[i] == '/'){
                pushfl(1/((1/popfl())+(1/popfl())));

            }
            ++i;
        }
        else{
            if (postfix[i] != ' '){
                while(isalnum(postfix[i])){
                    sub[j++] = postfix[i];
                    i++;
                }
                pushfl((float) atof(sub));
                memset(sub,0,sizeof(sub));
                j=0;

            }
            ++i;
        }

    }
}

void infix_to_postfix(){
    int i =0,j=0,k=0;
```

```
151     char sub[max];
152     while(infix[i] != '\0'){
153         if (isalnum(infix[i])){
154             while(isalnum(infix[i])){
155                 postfix[k++] = infix[i++];
156             }
157             postfix[k++] = ' ';
158         }
159 
160         else if (infix[i] == '('){
161             push(infix[i]);
162             i++;
163         }
164         else if(infix[i] == ')'){
165             while(stack[tos] != '('){
166                 postfix[k++] = pop();
167             }pop();
168             i++;
169         }
170         else{
171             if (tos == -1){
172                 push(infix[i]);
173             }
174             else{
175                 while(prec(stack[tos]) >= prec(infix[i])){
176                     postfix[k++] = pop();
177                 }
178                 push(infix[i]);
179             }
180             i++;
181         }
182 
183     }
184     while(tos != -1){
185         postfix[k++] = pop();
186     }
187     postfix[k++] ='\0';
188 }
189 
190 main(){
191     int i=0,b;
192     printf("\n\n====================EQUIVALENT RESISTANCE
    CALCULATOR====================\n\n");
193     printf("Use S for series and P for parallel\n");
194     printf("For example: a 12 ohm resistor in series with a parallel combination of 24 ohm
    and 12 ohm resistors is entered as 12S(24P12)\n");
195     printf("Enter the resistance expression:\t");
196     gets(infix);
197 
198     while(infix[i] != '\0'){
```

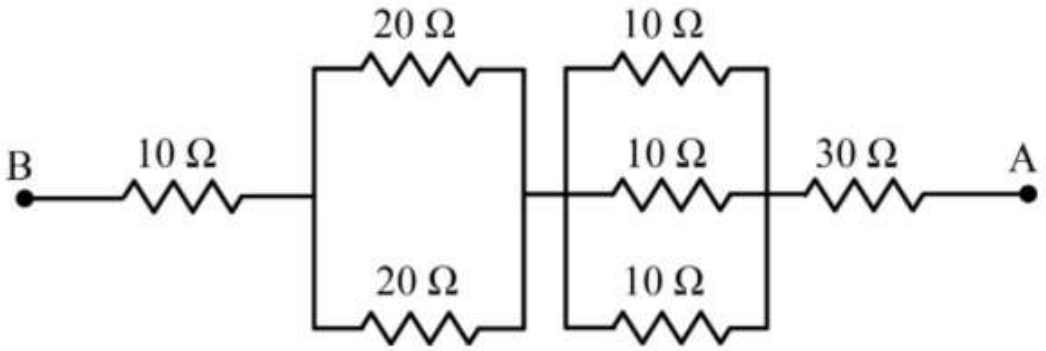
```
        if (infix[i] == 'S' || infix[i] == 's'){
            infix[i] = '+';
        }
        if (infix[i] == 'P' || infix[i] == 'p'){
            infix[i] = '/';
        }
        i++;
    }
b = balanced();
if (b==1){
    infix_to_postfix();
evalpost();
printf("\nThe Equivalent resistance between the 2 points for the given circuit is %.2f
ohms\n",member[0]);
}
else{
    printf("Enter the equation correctly\n");
}

}

```

EXAMPLE :

Find the equivalent resistance between points A and B



Manual solution:

We can clearly see that two resistors 20 ohms are connected in parallel (see figure below). We can replace them with one equivalent resistor, the resistance of which can be calculated from the following expression

$$\frac{1}{R_{eq}} = \frac{1}{R_1} + \frac{1}{R_2} = \frac{1}{20} + \frac{1}{20} = \frac{1}{10}$$

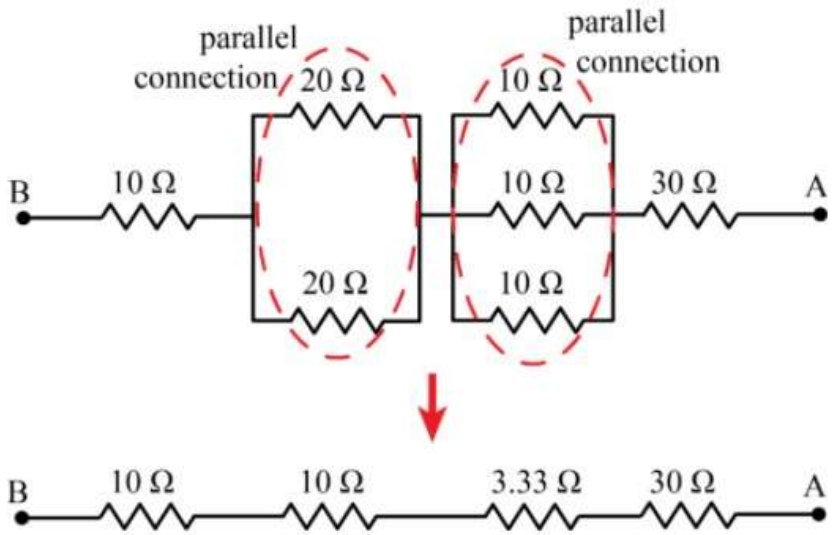
Then

$$R_{eq} = 10\ \Omega$$

We also have three resistors 10 ohm, 10 ohm, and 10 ohm connected in parallel (see figure below). We replace them with an equivalent resistance of

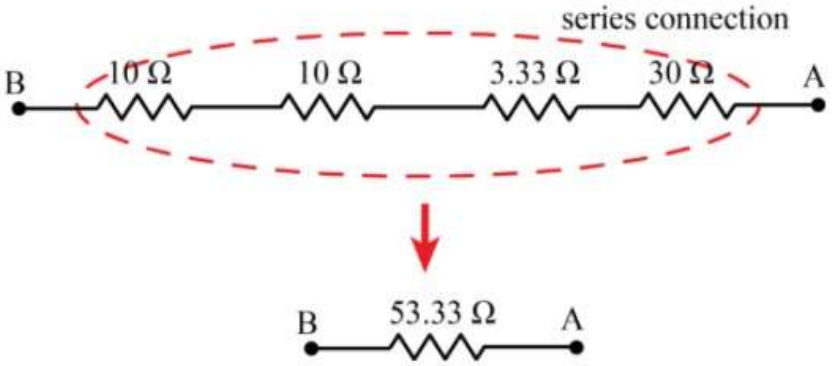
$$\frac{1}{R_{eq}} = \frac{1}{R_1} + \frac{1}{R_2} + \frac{1}{R_3} = \frac{1}{10} + \frac{1}{10} + \frac{1}{10} = \frac{3}{10}$$

$$R_{eq} = 3.33\ \Omega$$



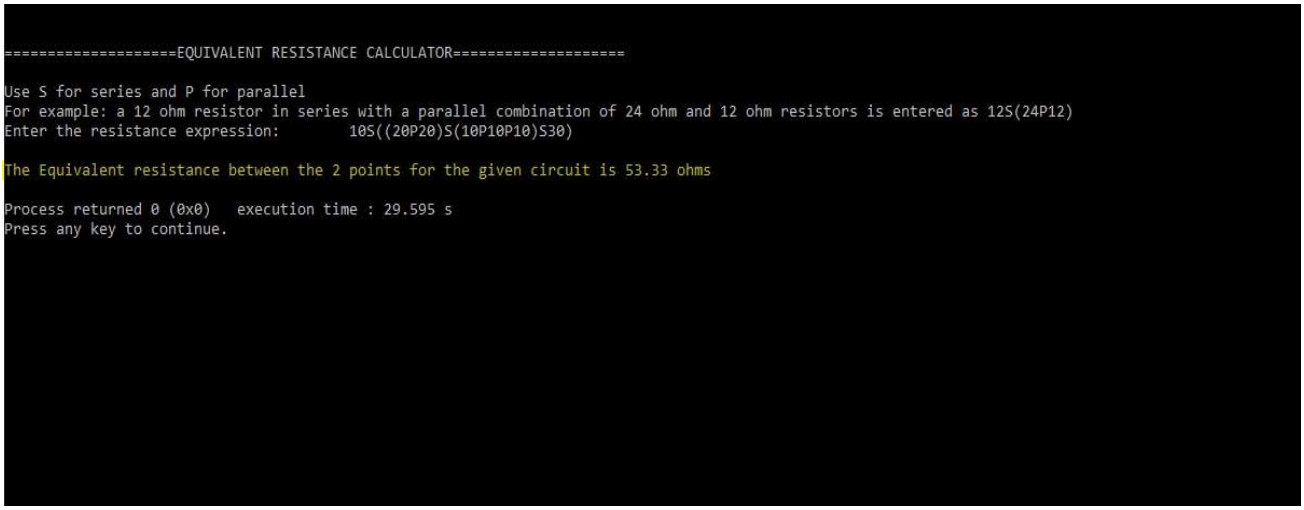
Then we have four resistors (10 ohm, 10 ohm, 3.33 ohm, and 30 ohm) connected in series, see figure below. The equivalent resistance of these resistors can be found from the following expression

$$R_{eq} = R_1 + R_2 + R_3 + R_4 = 10 + 10 + 3.33 + 30 = 53.33\ \Omega$$



Solution from code:

Equivalent resistance expression: 10S((20P20)S(10P10P10)S30)



Answer:

We can see from the 2 results the equivalent resistance between points A and B is 53.33 ohms

INFERENCE:

The ohm meter is thus designed digitally(equivalent resistance between two points is found).

REFERENCES:

For examples:

https://physics.info/circuits-r/practice.shtml

http://www.solvephysics.com/topic_electricity_equivalent_resistance.shtml

For code:

Class notes.