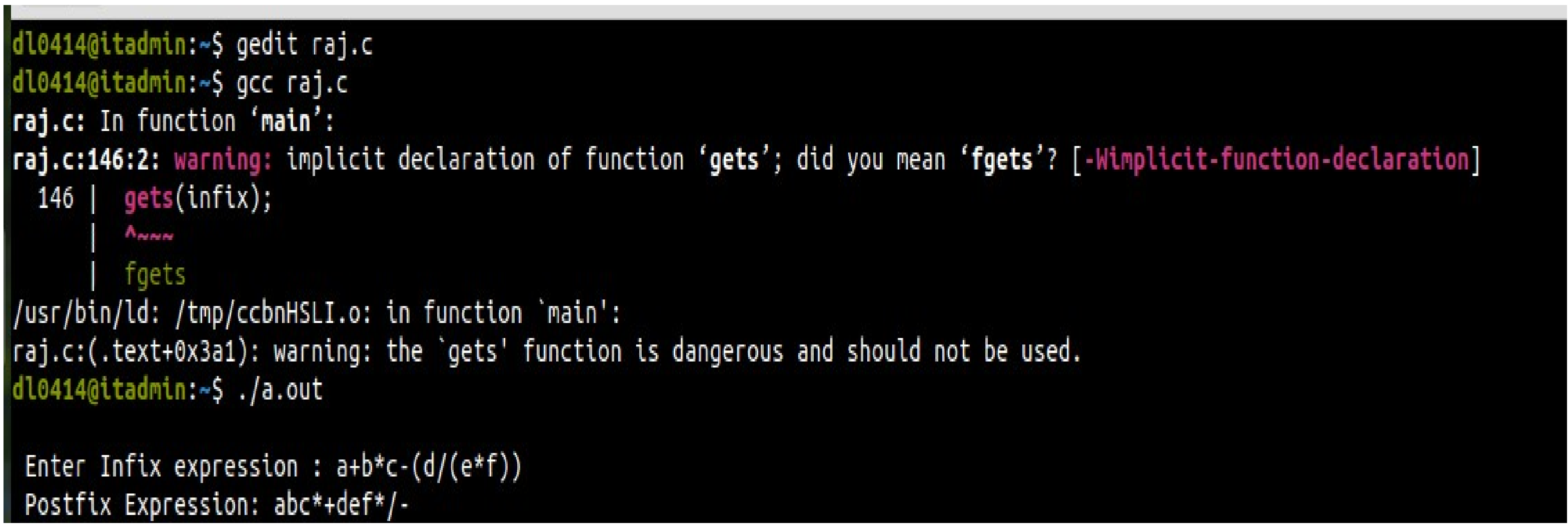
Experiment 3



```
dl0414@itadmin:~$ gedit raj.c
dl0414@itadmin:~$ gcc raj.c
raj.c: In function ‘main’:
raj.c:146:2: warning: implicit declaration of function ‘gets’; did you mean ‘fgets’? [-Wimplicit-function-declaration]
  146 |  gets(infix);
      |  ^~~~
      |  fgets
/usr/bin/ld: /tmp/ccbnHSLI.o: in function `main':
raj.c:(.text+0x3a1): warning: the `gets' function is dangerous and should not be used.
dl0414@itadmin:~$ ./a.out

 Enter Infix expression : a+b*c-(d/(e*f))
 Postfix Expression: abc*+def*/-
```

```
#include<stdio.h>
#include<stdlib.h>
#include<ctype.h>
#include<string.h>

#define SIZE 100
char stack[SIZE];
int top = -1;


void push(char item)
{
	if(top >= SIZE-1)
	{
		printf("\n Stack Overflow.");
	}
	else
	{
		top = top+1;
		stack[top] = item;
	}
}

char pop()
{
	char item ;

	if(top <0)
	{
		printf("stack under flow.");
		getchar();
		exit(1);
	}
	else
	{
```

```
                item = stack[top];
                top = top-1;
                return(item);
        }
}

int is_operator(char symbol)
{
        if(symbol == '^' || symbol == '*' || symbol == '/' || symbol == '+' || symbol =='-')
        {
                return 1;
        }
        else
        {
        return 0;
        }
}

int precedence(char symbol)
{
        if(symbol == '^')
        {
                return(3);
        }
        else if(symbol == '*' || symbol == '/')
        {
                return(2);
        }
        else if(symbol == '+' || symbol == '-')
        {
                return(1);
        }
        else
        {
                return(0);
        }
}

void InfixToPostfix(char infix_exp[], char postfix_exp[])
{
        int i, j;
        char item;
        char x;

        push('(');
        strcat(infix_exp,")");

        i=0;
        j=0;
        item=infix_exp[i];

        while(item != '\0')
```

```
{
        if(item == '(')
        {
                push(item);
        }
        else if( isdigit(item) || isalpha(item))
        {
                postfix_exp[j] = item;
                j++;
        }
        else if(is_operator(item) == 1)
        {
                x=pop();
                while(is_operator(x) == 1 && precedence(x)>= precedence(item))
                {
                        postfix_exp[j] = x;
                        j++;
                        x = pop();
                }
                push(x);

                push(item);
        }
        else if(item == ')')
        {
                x = pop();
                while(x != '(')
                {
                        postfix_exp[j] = x;
                        j++;
                        x = pop();
                }
        }
        else
        {
                printf("\nInvalid infix Expression.\n");
                getchar();
                exit(1);
        }
        i++;


        item = infix_exp[i];
}
if(top>0)
{
        printf("\nInvalid infix Expression.\n");
        getchar();
        exit(1);
}

postfix_exp[j] = '\0';
```

```
	}
int main()
{
	char infix[SIZE], postfix[SIZE];

	printf("\n Enter Infix expression : ");
	gets(infix);

	InfixToPostfix(infix,postfix);
	printf(" Postfix Expression: ");
	puts(postfix);

	return 0;
}
```