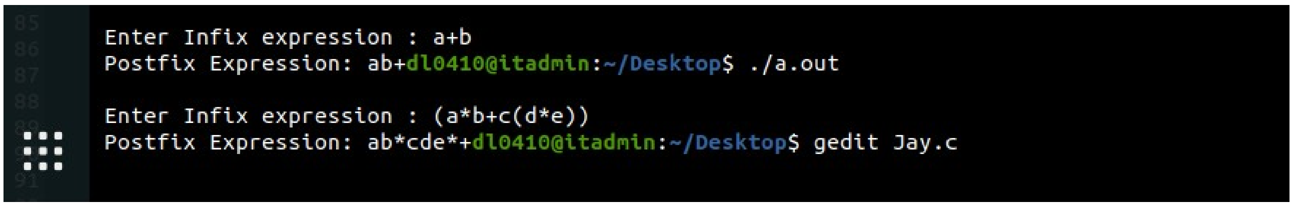
Experiment 3

Code:

```
#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: invalid infix expression");
	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 : ");
        scanf("%s",infix);
        InfixToPostfix(infix,postfix);
        printf(" Postfix Expression: ");
        printf("%s",postfix);
        return 0;
}
```

Output:



```
Enter Infix expression : a+b
Postfix Expression: ab+dl0410@itadmin:~/Desktop$ ./a.out

Enter Infix expression : (a*b+c(d*e))
Postfix Expression: ab*cde*+dl0410@itadmin:~/Desktop$ gedit Jay.c
```