HW Assignment 3

Marshall Slagle
7/13/18

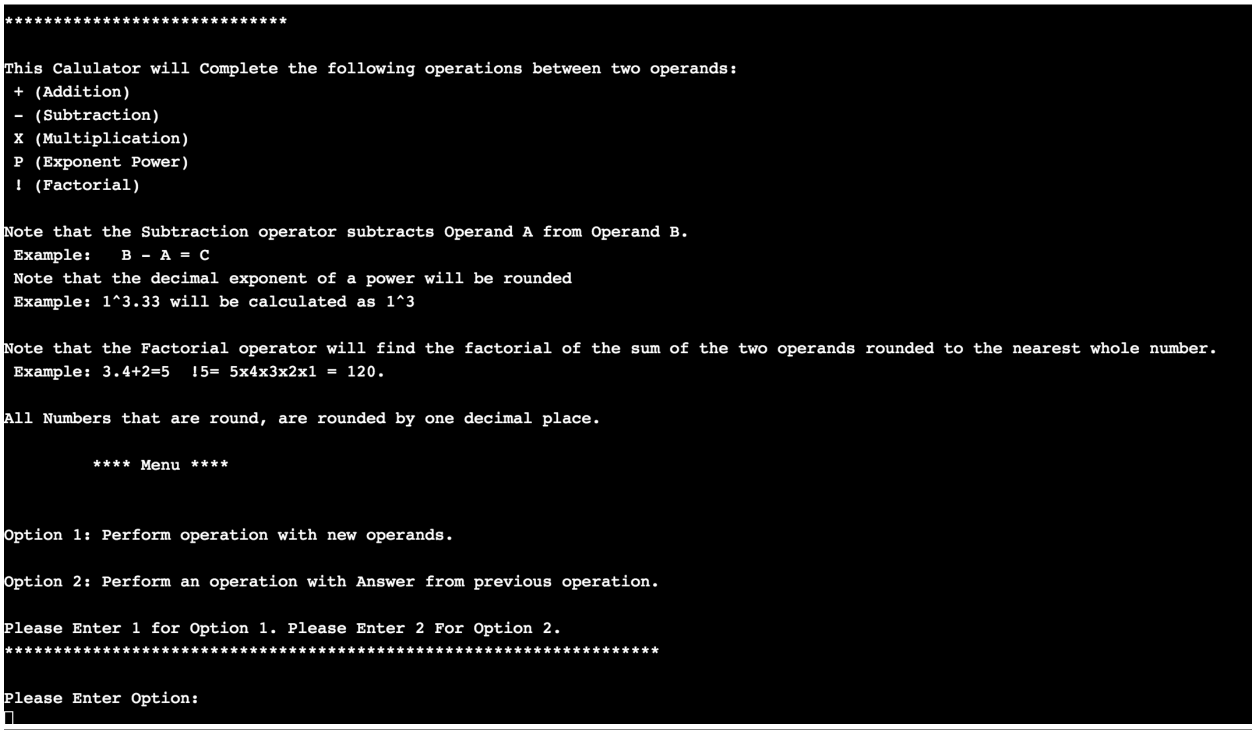
Program Code

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


void CalInput(float *oA, float *oB, float *pA, char *oP, int userMenuInput); //Method
Declaration
float OperationChooser1(char operator, float operandA, float operandB); //Method
Declaration
float OperationChooser2(char operator, float operandA, float prevAnswer); //Method
Declaration
float Addition(float operandA, float operandB); //Method Declaration
float Subtraction(float operandA, float operandB); //Method Declaration
float Multiplication(float operandA, float operandB); //Method Declaration
float Power(float operandA, float operandB); //Method Declaration
float Factorial(float operandA, float operandB); //Method Declaration




void CalInput(float *oA, float *oB, float *pA, char *oP, int userMenuInput) //Method
retrieves operands and operator from user
{
    if(userMenuInput==1) //If to run new calculation option passed from paramenter
    {
        printf("\nPlease Enter Operand A: \n"); //display message
        scanf("%f", &*oA); //retrieves operand A from user
        printf("\nPlease Enter Operand B: \n"); //display message
        scanf("%f", &*oB); //retrieves operand B from user
        printf("\nPlease Enter The Operator: \n"); //display message
        int i=1; //initializing counter for operator input loop
        while(i!=0) //input loop used to require user to input acceptable operator
        {
            scanf(" %c", &*oP); //retrieves operator as character from user. space is
used to prevent misrecognition

            if(*oP=='+' || *oP=='-' || *oP=='x' || *oP=='X' || *oP=='p' || *oP=='P' ||
*oP=='!') //if statement to determine if user input is acceptable
            {
                i=0; //if true loop will end.

            }
            else //if not true
            {
                printf("\nThe Operator entered is not a valid option for this
Calculator. Please enter a valid Operator.\n"); //error message to prompt for new
input
                i++; //continues loop
            }
        }

    }
    if(userMenuInput==2) //If to run calculation option with previous answer passed
from paramenter
    {
        printf("\nPlease Enter Operand A: \n"); //prompt message
        scanf("%f", &*oA); //retrieves operand A from user
        printf("\nThe Previous Answer will be used for Operand B.\n"); //message to
inform user that operand b will be replaced with the previously calculated answer
        int i=1; //initializing counter for operator input loop
        while(i!=0) //while loop to accept operater from user
        {
            printf("\nPlease Enter The Operator:\n"); //prompt message
            scanf(" %c", &*oP); //retrieves operator as character from user. space is
used to prevent misrecognition
            if(*oP=='+' || *oP=='-' || *oP=='x' || *oP=='X' || *oP=='p' || *oP=='P' ||
*oP=='!') //used to determind if character entered and assigned to pointer variabel
destination is a valid operator for calculator
            {
                i=0; //ends loop if true

            }
            else //else to display error if incorrect
            {
                printf("\nThe Operator entered is not a valid option for this
Calculator. Please enter a valid Operator.\n"); //error message
                i++; //continues loop since operator is invalid
            }
        }
    }


}

float OperationChooser1(char operator, float operandA, float operandB) //intermediate
method used to call calculation methods and pass parameters to the correct method
based on new operands option selected
{
    float finalCalc; //declares final calculation

    if(operator=='+') //if statement to call addition method if character matches
    {
        finalCalc = Addition(operandA, operandB); //calls addition method and sends
two parameters, while retrieving calculation from method
        return finalCalc; //returns final calculation to main method
    }
    if(operator=='-') //if statement to call subtraction method if character matches
    {
        finalCalc = Subtraction(operandA, operandB); //calls subtraction method and
sends two parameters, while retrieving calculation from method
        return finalCalc; //returns final calculation to main method
    }
    if(operator=='x' || operator=='X') //if statement to call multiplication method if
character matches
    {
        finalCalc = Multiplication(operandA, operandB); //calls Multiplication method
and sends two parameters, while retrieving calculation from method
        return finalCalc; //returns final calculation to main method
    }
    if(operator=='p' || operator=='P') //if statement to call power method if
character matches
    {
        finalCalc = Power(operandA, operandB); //calls Power method and sends two
parameters, while retrieving calculation from method
        return finalCalc; //returns final calculation to main method
    }
    if(operator=='!') //if statement to call factorial method if character matches
    {
        finalCalc = Factorial(operandA, operandB); //calls factorial method and sends
two parameters, while retrieving calculation from method
        return finalCalc; //returns final calculation to main method
    }
    return -1; //returns -1 if no if statement is true; however, this should only
happen out of error, since the characters are checked in a previous method before
being sent
}

float OperationChooser2(char operator, float operandA, float prevAnswer)
//intermediate method used to call calculation methods and pass parameters to the
correct method based on character entered and prevanswer option selected
{
    float finalCalc; //declares final calculation

    if(operator=='+') //if statement to call addition method if character matches
    {
        finalCalc = Addition(operandA, prevAnswer); //calls addition method and sends
two parameters, while retrieving calculation from method
        return finalCalc; //returns final calculation to main method
    }
    if(operator=='-') //if statement to call subtraction method if character matches
    {
        finalCalc = Subtraction(operandA, prevAnswer); //calls subtraction method and 
sends two parameters, while retrieving calculation from method
        return finalCalc; //returns final calculation to main method
    }
    if(operator=='x' || operator=='X') //if statement to call multiplication method if 
character matches
    {
        finalCalc = Multiplication(operandA, prevAnswer); //calls Multiplication 
method and sends two parameters, while retrieving calculation from method
        return finalCalc; //returns final calculation to main method
    }
    if(operator=='p' || operator=='P') //if statement to call power method if 
character matches
    {
        finalCalc = Power(operandA, prevAnswer); //calls Power method and sends two 
parameters, while retrieving calculation from method
        return finalCalc; //returns final calculation to main method
    }
    if(operator=='!') //if statement to call factorial method if character matches
    {
        finalCalc = Factorial(operandA, prevAnswer); //calls factorial method and 
sends two parameters, while retrieving calculation from method
        return finalCalc; //returns final calculation to main method
    }
    return -1; //returns -1 if no if statement is true; however, this should only 
happen out of error, since the characters are checked in a previous method before 
being sent
}


float Addition(float operandA, float operandB) //method to find the sum of two 
operands
{
    return operandA+operandB; //returns the sum of two operands
}

float Subtraction(float operandA, float operandB) //method to find the subtraction of 
two operands
{
    return operandB-operandA; //returns the subtraction of an operand from an operand
}


float Multiplication(float operandA, float operandB) //method to find an operand times 
an operand
{
    return operandA*operandB; //returns the two operands times together
```

```
}

float Power(float operandA, float operandB) //method to calculate the power of a
number given an exponent
{

    float positiveExponent; //declares positive exponent value, used to store positive
exponents
    float decimal = 0.5; //decimal variable used to round decimal of exponent to next
whole number befor being truncated
    int finalExponent; //declaring variable to final exponent after truncating
decimals
    float answer; //declaring variable for final answer

if(operandA<0 && operandB<0) //if both operand are negative will print error
    {
        printf("\n **** Out of Bounds ***** \n"); //error message
        return -1; //returns -1 for error message
    }


    if(operandB<0 && operandB != -1) //if statement for if original exponent is less
then 0 and not equal to -1
    {
        positiveExponent = 0-operandB; //creates absolute value of negative variable
        positiveExponent = positiveExponent + decimal; //adds decimal value to
positive exponent to round before truncated
        finalExponent = (int)positiveExponent; //truncating variable

        answer = operandA; //used to set answer to value of operand to first exponent
value of 1 for loop
        for(int i = 1; i < finalExponent; i++) //used to loop power calculation until
1 iteration before entered exponent, since a first iteration is already established
        {
            answer = answer * operandA; //times each iteration by original operand
        }
        answer = 1/answer; //divides 1 by answer to find negetive exponent of power
        return answer; //returns answer to previous method

    }
    if(operandB==0) //if statement for if original exponent is equal to 0
    {
        answer=1; //setting answer to 1, since all powers to the exponent of 0 equals
1
        return answer; //returning answer to previous method
    }
    if(operandB==-1) //if statement for if original exponent is less then 1
    {
        answer=operandA; //setting answer to operand, since the exponent of 1 is the
operand given before solving for negative
        return 1/answer; //returning answer of negative exponent to previous method
    }
    if(operandB>1) //if statement for if original exponent is greater then 1
    {
        positiveExponent = operandB; //setting positive exponent variable to operand
        positiveExponent = positiveExponent + decimal; //adds decimal value to
positive exponent to round before truncated
        finalExponent = (int)positiveExponent; //truncating variable

        answer = operandA; //used to set answer to value of operand to first exponent
value of 1 for loop
        for(int i = 1; i < finalExponent; i++) //used to loop power calculation until
1 iteration before entered exponent, since a first iteration is already established
        {
            answer = answer * operandA; //times each iteration by original operand
        }
        return answer; //returning answer to previous method
    }
    else //used to find exponent of 1, which is single iteration of final answer
equaling operand, since exponent equal to 1 is the only other exponent option not
covered by the if statements
    {
        answer=operandA; //setting answer to operand
        return answer; //returning answer to previous method
    }



}


float Factorial(float operandA, float operandB) //Method to calculate factorial value
{

    operandA = operandA + 0.5; //rounds operandA by adding 0.5 to the value. If
original value has a decimal place of 0.5 or greater it will bring the number to the
next whole number before having the decimals truncated
    operandB = operandB + 0.5; //rounds operandB by adding 0.5 to the value. If
original value has a decimal place of 0.5 or greater it will bring the number to the
next whole number before having the decimals truncated
    int sum = (int)operandA + (int)operandB; //creates an integer sum of the rounded
operands with the decimals truncated

    if(sum > 0) //if statement to determine if sum is greater then 0
    {
        int factorl = sum; //sets factorl variable used for calculation in loop to 
sum, inorder to be set to first iteration of loop, since loop runs in decending order 
to calculate factorial

        for(int i = sum -1; i > 1; i--) //for loop to run the factorial for all 
iterations of the factorial exept the begining, since it is prevously set to the first 
iteration. And last iteration since anything times 1 is still the same
        {
           factorl = factorl * i; //calulates the factorial of each iteration
        }

        return (float)factorl; //returns the factorial of the two operands
    }
    if(sum == 0) //if the sum is equal to 0
    {
        return 1.0; //since the factorial of 0 is 1 the program will return 1 to the 
previous method
    }
    else //if sum is less then 0
    {
        printf("\n **** Such operation is invalid. Operands out of bounds. ****\n"); 
//prints error message
        return -1; //returns -1 if sum of factorial is less then 0
    }

}





int main() //main method
{

int userMenuInput; //declare user input for menu variable
char userReplay; //declare user replay for loop variable
float operandA; //declare operand a variable
float operandB; //declare operand b variable
float newAnswer; //declare new anser variable to store answer after current 
calculation
float prevAnswer = 0; //initializing previous answer variable to 0
char operator; //declare operator variable.

printf("*****************************\n"); //display border
printf("\nWelcome to the Calculator\n"); //display welcome message
printf("\n*****************************\n"); //display border
```

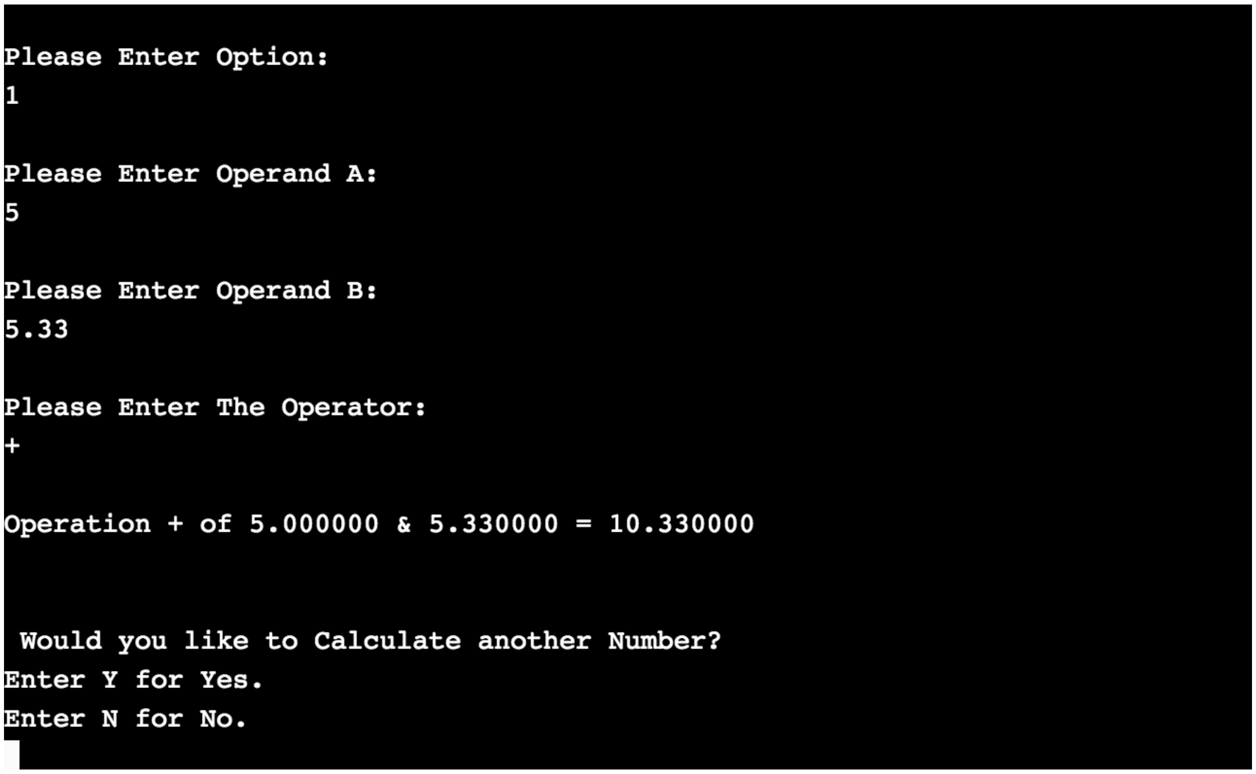
```
int i=1; //initializing variable counter to 1
while(i!=0) //while loop to keep base menu running until user inputs change
{
    printf("\nThis Calulator will Complete the following operations between two 
operands:\n + (Addition)\n - (Subtraction)\n X (Multiplication)\n P (Exponent Power)\n 
! (Factorial)\n"); //diplay instructions
    printf("\nNote that the Subtraction operator subtracts Operand A from Operand B.\n 
Example:   B - A = C\n Note that the decimal exponent of a power will be rounded\n 
Example: 1^3.33 will be calculated as 1^3\n"); //diplay instructions
    printf("\nNote that the Factorial operator will find the factorial of the sum of 
the two operands rounded to the nearest whole number.\n Example: 3.4+2=5  !5= 
5x4x3x2x1 = 120.\n"); //diplay instructions
    printf("\nAll Numbers that are round, are rounded by one decimal place.\n"); 
//diplay instructions
    printf("\n         **** Menu ****         \n\n"); //display border
    printf("\nOption 1: Perform operation with new operands.\n"); //diplay 
instructions
    printf("\nOption 2: Perform an operation with Answer from previous operation.\n"); 
//diplay instructions
    printf("\nPlease Enter 1 for Option 1. Please Enter 2 For Option 2."); //diplay 
instructions
    printf("\n*******************************************************************\n"); 
//display border
    printf("\nPlease Enter Option: \n"); //prompt for user input
    scanf("%d",&userMenuInput); //retreives user input

    switch(userMenuInput) //switch statement to determine which type of operands will 
be used in calculations
    {
        case 1: //if user wants new operands
            CalInput(&operandA, &operandB, &prevAnswer, &operator, userMenuInput); 
//sends user input variables to method by reference with pointers to retrieve user 
input
            newAnswer = OperationChooser1(operator, operandA, operandB); //calls 
intermediate method while passing parameters and retrieves final calculation
            printf("\nOperation %c of %f & %f = %f\n",operator, operandA, operandB, 
newAnswer); //prints calculation with original user input
            prevAnswer = newAnswer; //sets previous answer to answer just calculated
            break; //breaks switch statement
        case 2: //if user wants to user previous answer choice as operand B
            CalInput(&operandA, &operandB, &prevAnswer, &operator, userMenuInput); 
//sends user input variables to method by reference with pointers to retrieve user 
input
            newAnswer = OperationChooser2(operator, operandA, prevAnswer); //calls 
intermediate method while passing parameters and retrieves final calculation
            printf("\nOperation %c of %f & %f = %f\n",operator, operandA, prevAnswer, 
newAnswer); //prints calculation with original user input
            prevAnswer = newAnswer; //sets previous answer to answer just calculated
            break; //breaks from switch case
        default: //default else statement displays error message
            printf("\n\n ** Incorrect Option entered. Please enter valid option. **
\n\n"); //error message
    }

    int j=1; //initializing variable counter to 1
    while(j!=0) //while loop to determine if user will end or continue using
calculator
    {
        printf("\n\n Would you like to Calculate another Number? \nEnter Y for Yes.
\nEnter N for No.\n"); //Displays options to reboot or discontinue calculator
operations
        scanf(" %c", &userReplay); //retrieves operator as character from user. space
is used to prevent misrecognition
        if(userReplay=='N' || userReplay=='n') //determines if character entered
equals no.
        {
            i = 0; //To make outerloop end when inner loop ends
            break; //ends innerlopp
        }
        if(userReplay=='Y' || userReplay=='y') //determines if character entered
equals yes.
        {
            i++; //continues outerloop
            j=0; //ends innerloop
        }
        else //if character is not found else statement displayes error message
        {
            printf("\nThe Character entered was not a correct option. Please Enter a
valid option.\n"); //displays error message
            j++; //continues inner loop, until user enters valid character option for
menu.
        }
    }

}

printf("\n\n ********* You Have Exited The Calculator ********* \n\n"); //message
establishing the calculator has ended


}
```

Program Screenshot Output



```
*****************************

This Calulator will Complete the following operations between two operands:
 + (Addition)
 - (Subtraction)
 X (Multiplication)
 P (Exponent Power)
 ! (Factorial)

Note that the Subtraction operator subtracts Operand A from Operand B.
 Example:   B - A = C
 Note that the decimal exponent of a power will be rounded
 Example: 1^3.33 will be calculated as 1^3

Note that the Factorial operator will find the factorial of the sum of the two operands rounded to the nearest whole number.
 Example: 3.4+2=5  !5= 5x4x3x2x1 = 120.

All Numbers that are round, are rounded by one decimal place.

         **** Menu ****


Option 1: Perform operation with new operands.

Option 2: Perform an operation with Answer from previous operation.

Please Enter 1 for Option 1. Please Enter 2 For Option 2.
*******************************************************************

Please Enter Option:
```

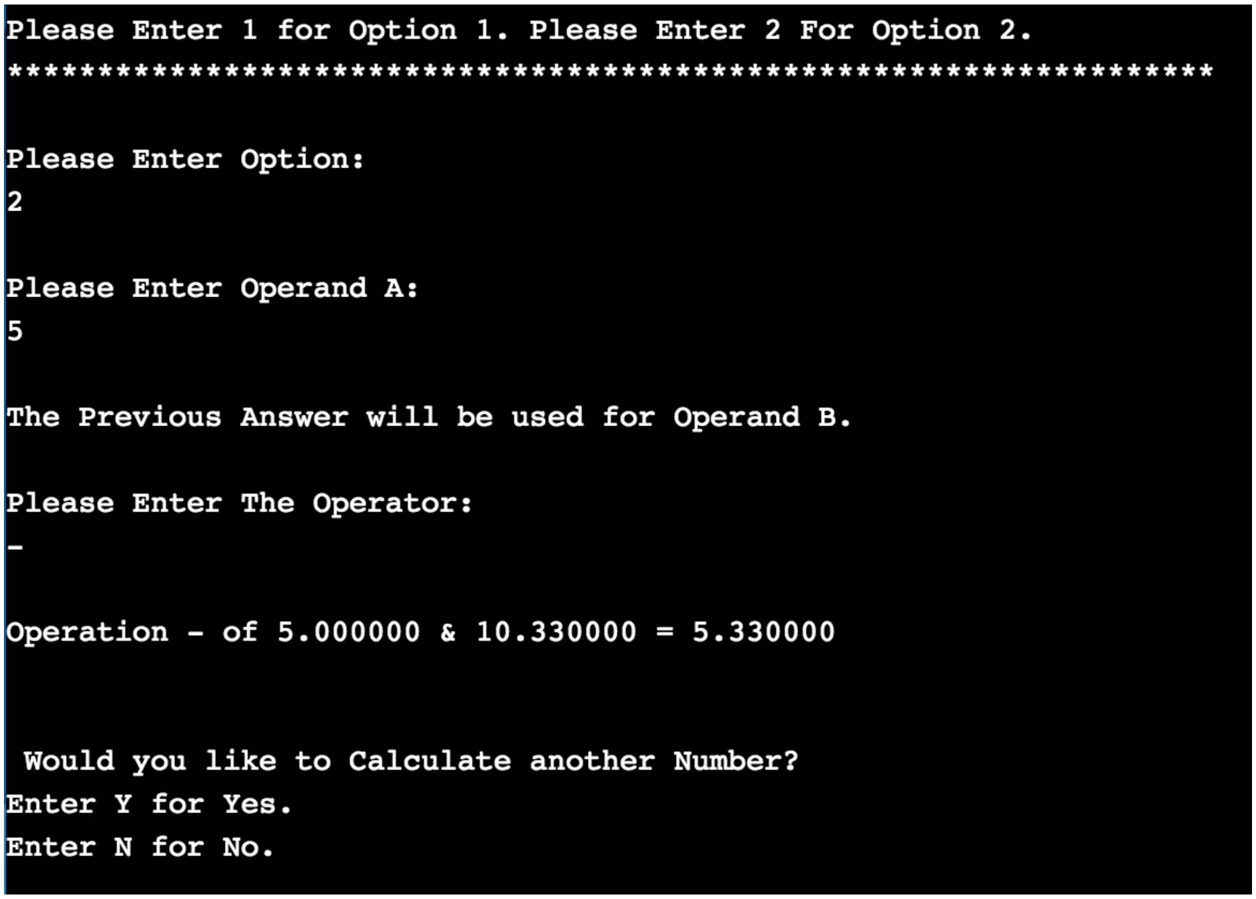
- Initial run.



```
Please Enter Option:
1

Please Enter Operand A:
5

Please Enter Operand B:
5.33

Please Enter The Operator:
+

Operation + of 5.000000 & 5.330000 = 10.330000


 Would you like to Calculate another Number?
Enter Y for Yes.
Enter N for No.
```

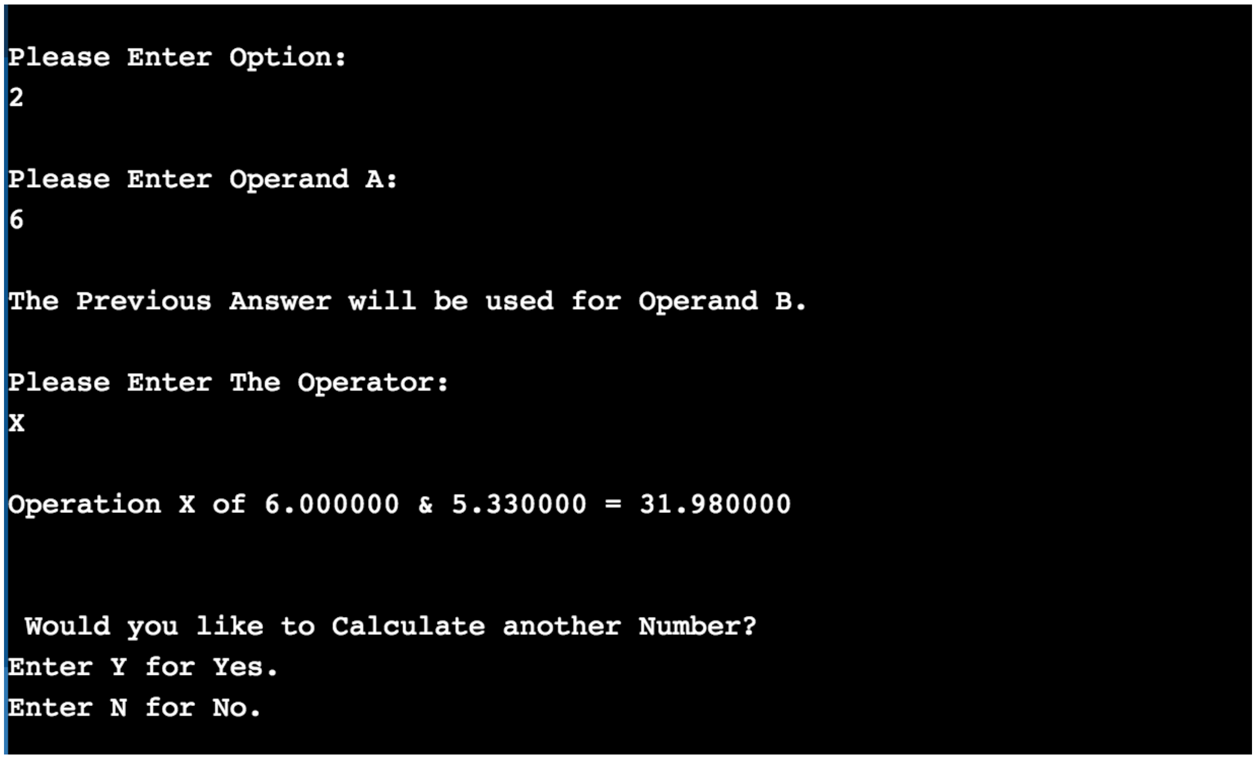
- Entered option 1 for new calculation with two operands.

- Entered 5 as operand A and 5.33 as operand B.
- Entered ‘+’ for addition.
- Output answer of 10.33
- Entered ‘Y’ for yes



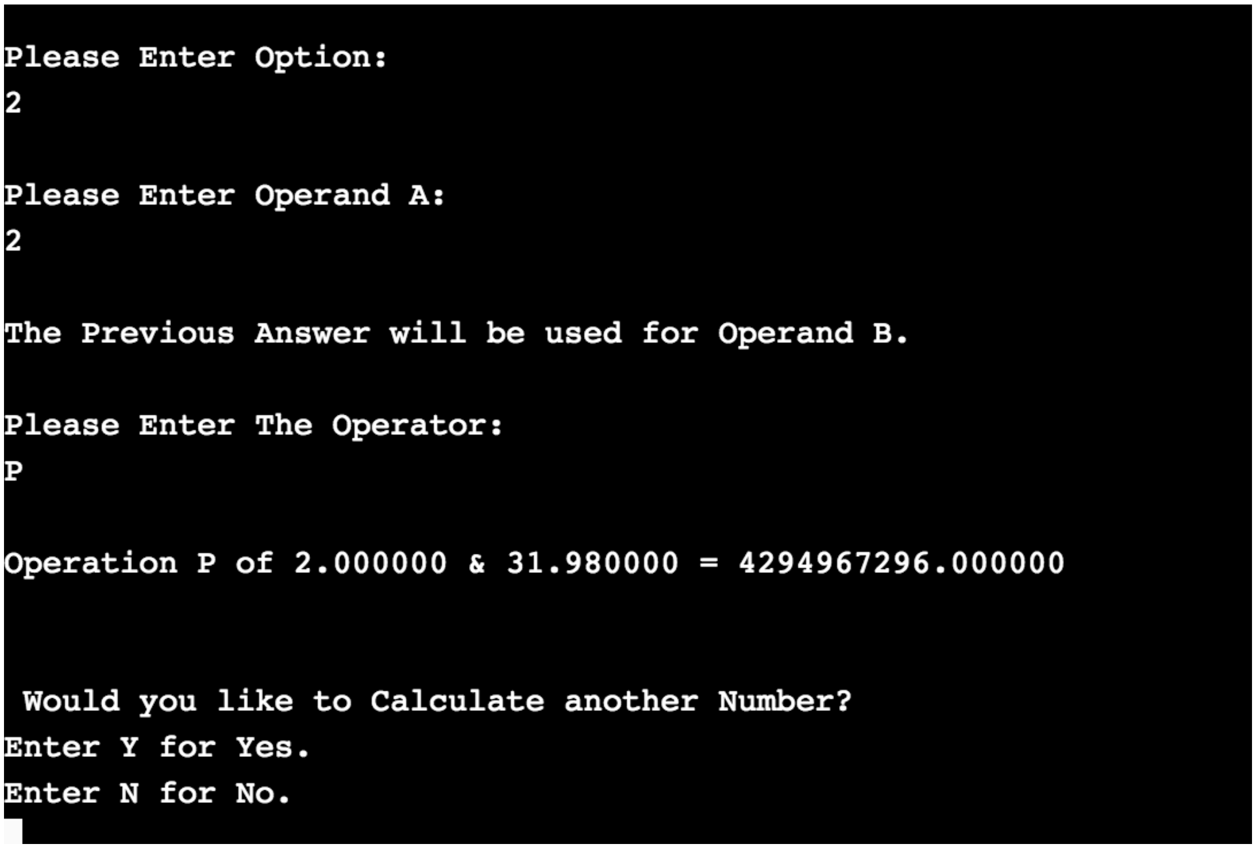
```
Please Enter 1 for Option 1. Please Enter 2 For Option 2.
******************************************************************

Please Enter Option:
2

Please Enter Operand A:
5

The Previous Answer will be used for Operand B.

Please Enter The Operator:
-

Operation - of 5.000000 & 10.330000 = 5.330000


 Would you like to Calculate another Number?
Enter Y for Yes.
Enter N for No.
```

- Selected option 2 for using previous answer as operand B.
- Entered 5 for operand A and entered ‘-‘ for operator.
- Answer is 5.33.
- Entered ‘Y’ for yes.



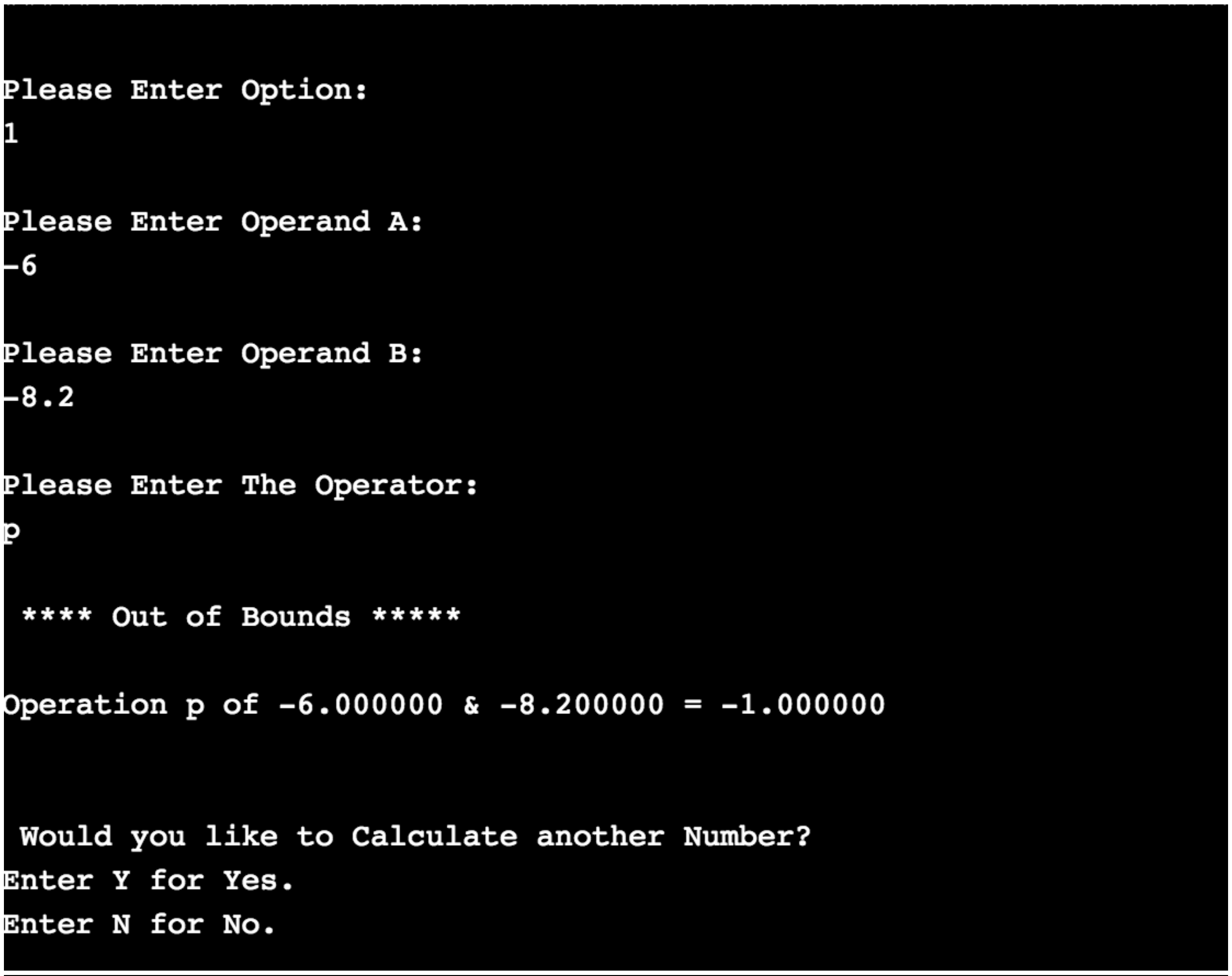
```
Please Enter Option:
2

Please Enter Operand A:
6

The Previous Answer will be used for Operand B.

Please Enter The Operator:
X

Operation X of 6.000000 & 5.330000 = 31.980000


 Would you like to Calculate another Number?
Enter Y for Yes.
Enter N for No.
```

- Selected option 2 for using previous answer for operand B.
- Entered 6 for operand A and entered ‘X’ for operator.
- Answer is 31.98
- Entered ‘Y’ for yes.



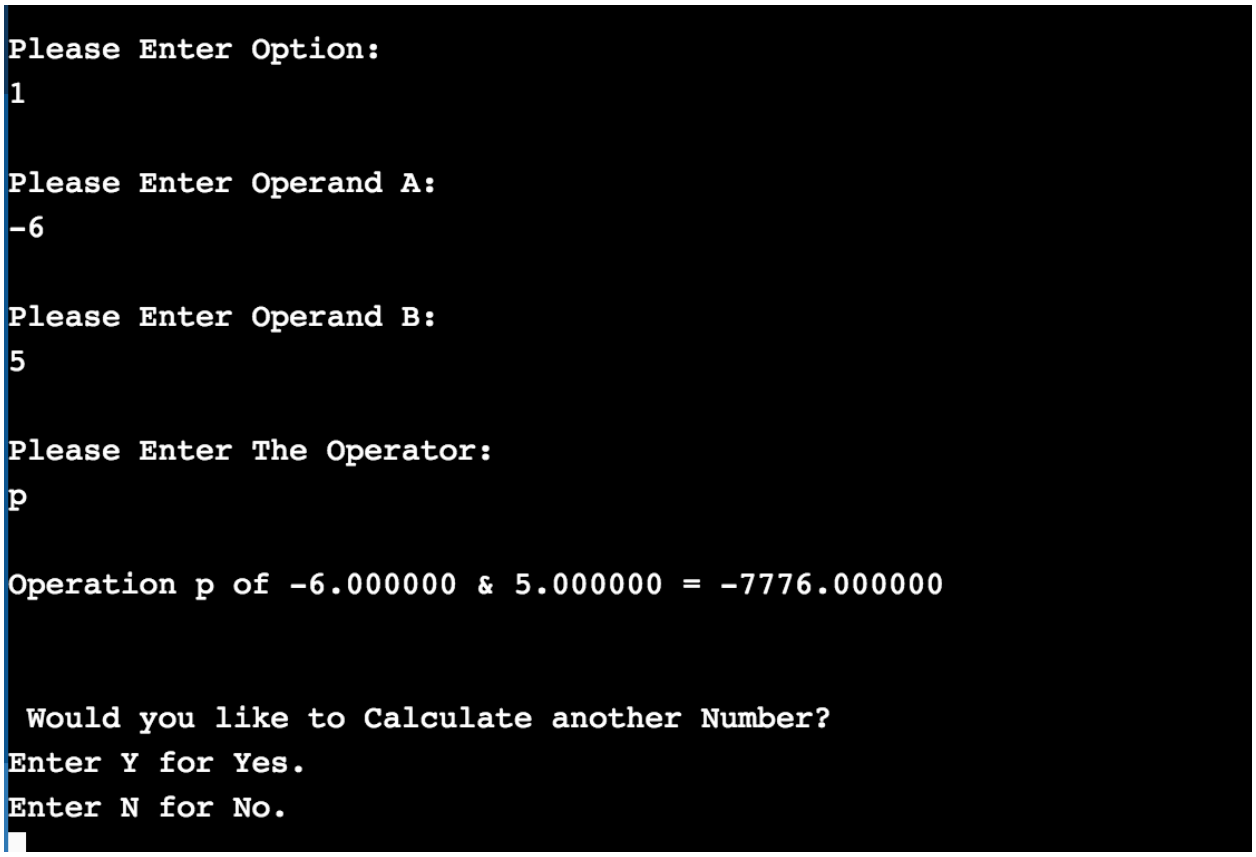
```
Please Enter Option:
2

Please Enter Operand A:
2

The Previous Answer will be used for Operand B.

Please Enter The Operator:
P

Operation P of 2.000000 & 31.980000 = 4294967296.000000



 Would you like to Calculate another Number?
Enter Y for Yes.
Enter N for No.
```

- Selected option 2 for using previous answer for operand B.
- Entered 2 for operand A and entered ‘P’ for operator.
- Answer for 2 to the power of 32(31.98 rounded) is 4294967296.
- Entered ‘Y’ for yes.



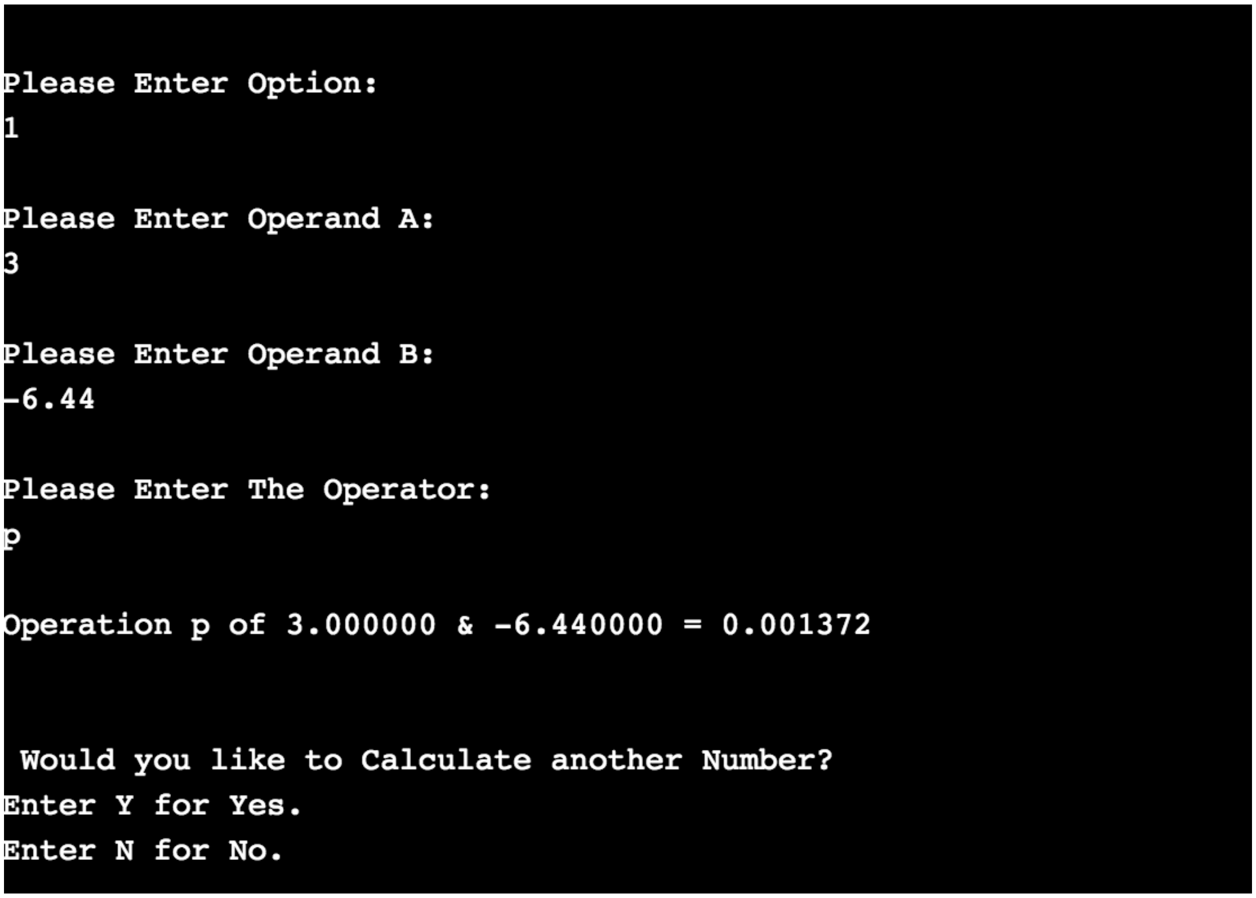
```
Please Enter Option:
1

Please Enter Operand A:
-6

Please Enter Operand B:
-8.2

Please Enter The Operator:
p

 **** Out of Bounds *****

Operation p of -6.000000 & -8.200000 = -1.000000


 Would you like to Calculate another Number?
Enter Y for Yes.
Enter N for No.
```

- Entered 1 for new operands.
- Entered -6 for A. and -8.2 for B.
- Entered ‘P’ for power.
- Error messages displayed since both operands are negative.

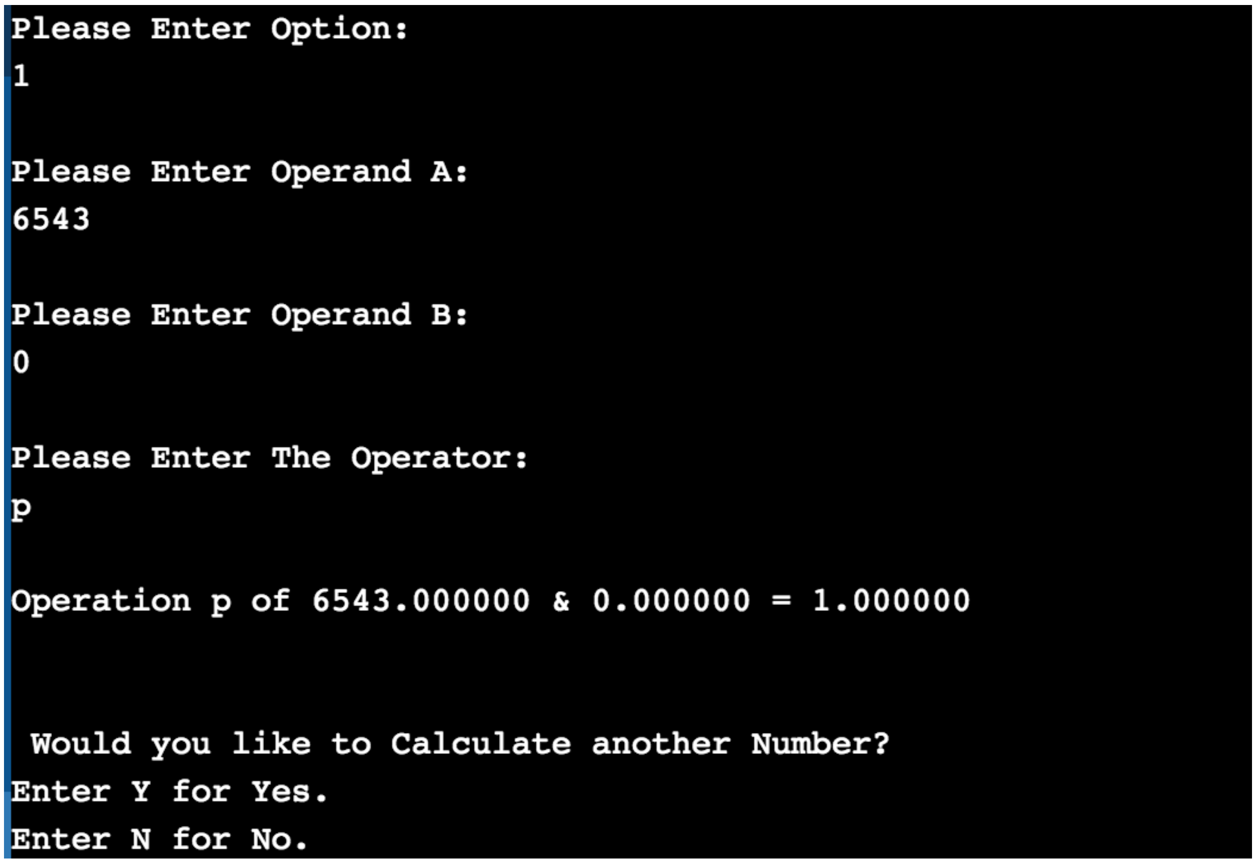


```
Please Enter Option:
1

Please Enter Operand A:
-6

Please Enter Operand B:
5

Please Enter The Operator:
p

Operation p of -6.000000 & 5.000000 = -7776.000000


 Would you like to Calculate another Number?
Enter Y for Yes.
Enter N for No.
```

- Selected option 1.
- Entered -6 for operand A.
- Entered 5 for operand B.
- Entered ‘P’ for Power operator.
- Correct output answer -7776.
- Entered ‘Y’ for yes.

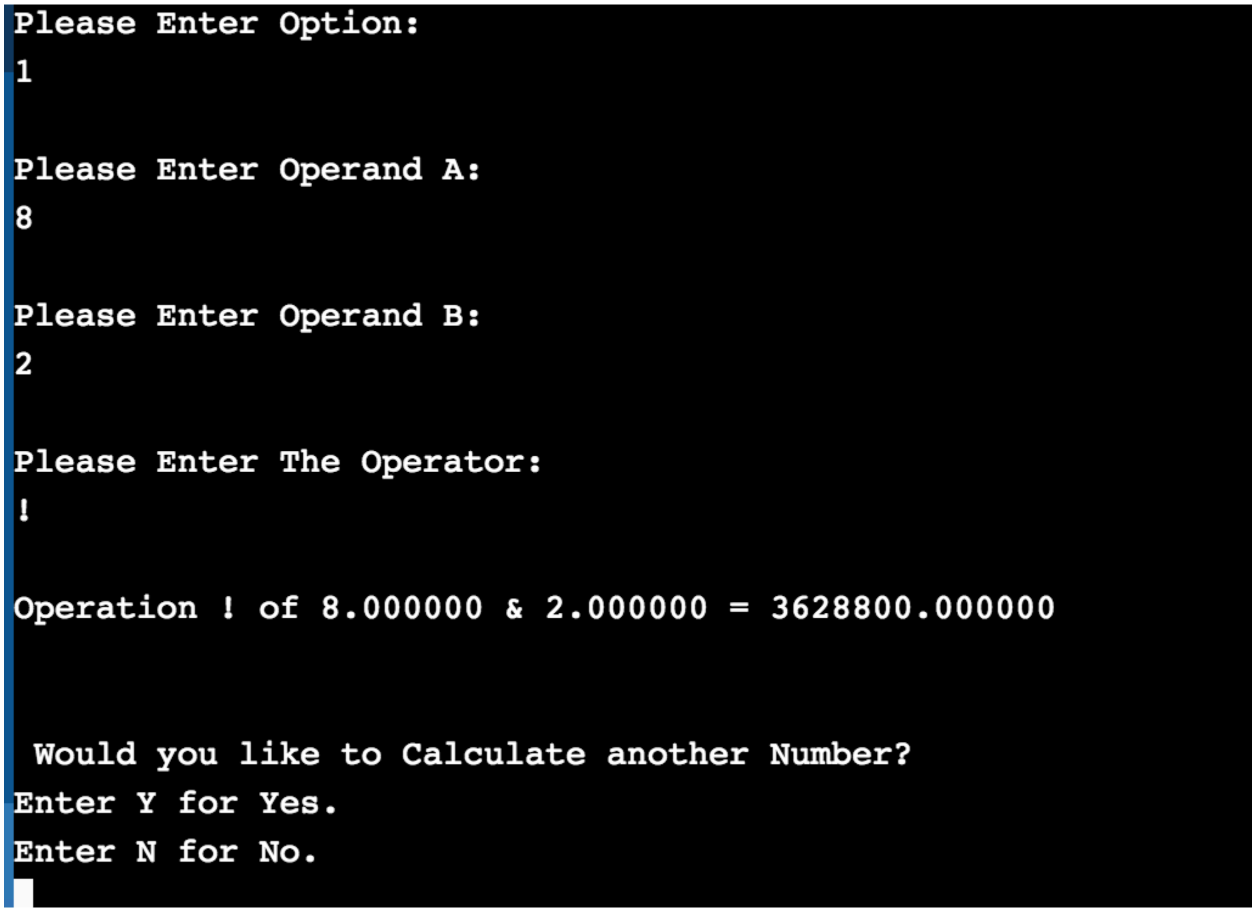

```
Please Enter Option:
1

Please Enter Operand A:
3

Please Enter Operand B:
-6.44

Please Enter The Operator:
p

Operation p of 3.000000 & -6.440000 = 0.001372


 Would you like to Calculate another Number?
Enter Y for Yes.
Enter N for No.
```

- Entered option 1.
- Entered 3 for operand A.
- Entered -6.44 for operand B.
- Entered ‘P’ for the Power operator.
- Correct output answer is 0.001372.
- Entered ‘Y’ for yes.



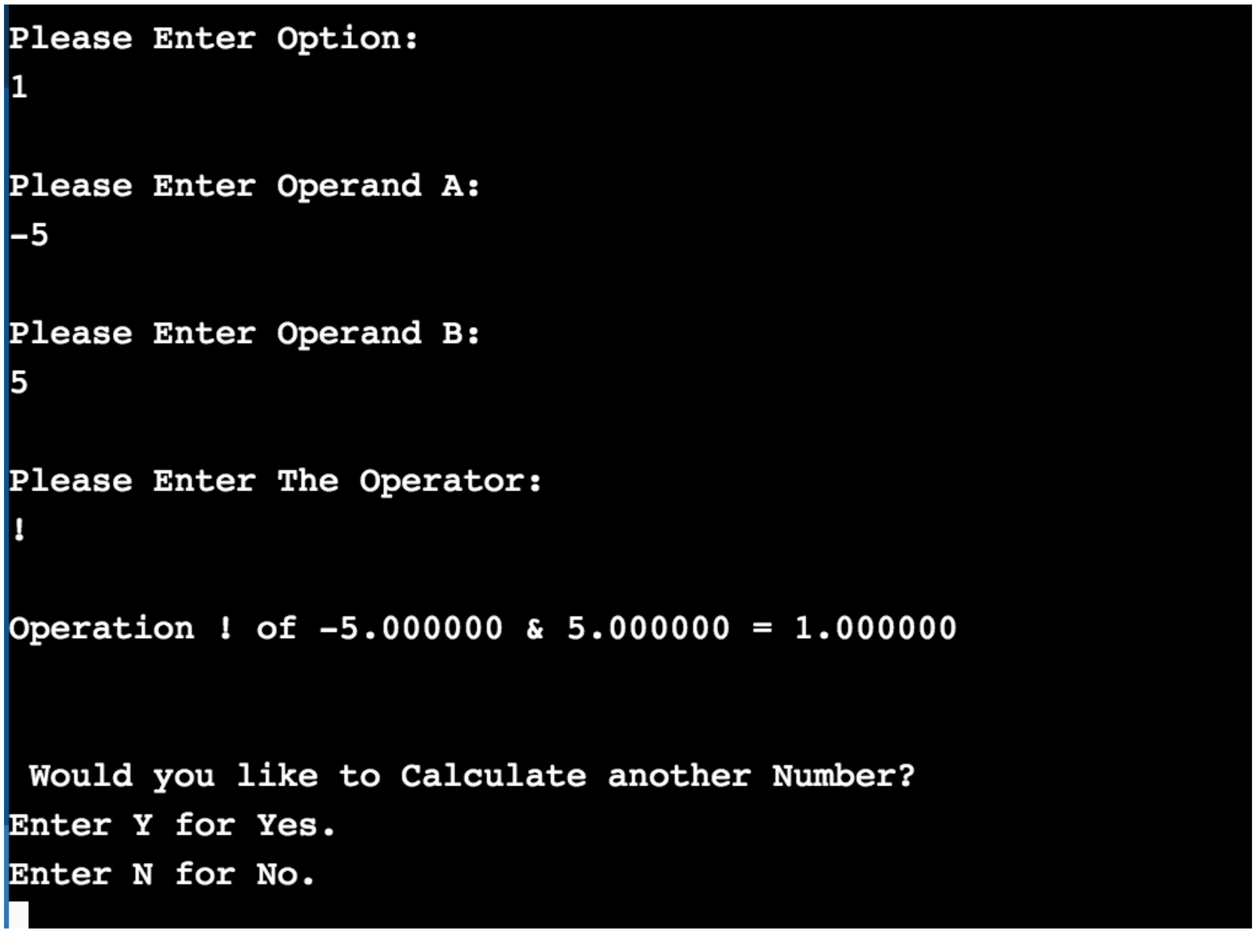
```
Please Enter Option:
1

Please Enter Operand A:
6543

Please Enter Operand B:
0

Please Enter The Operator:
p

Operation p of 6543.000000 & 0.000000 = 1.000000



 Would you like to Calculate another Number?
Enter Y for Yes.
Enter N for No.
```

- Entered option 1.
- Entered 6543 for Operand A.
- Entered 0 for Operand B.
- Correct output answer is 1, since all values to the exponent of 0 is 1.
- Entered ‘y’ for Yes.



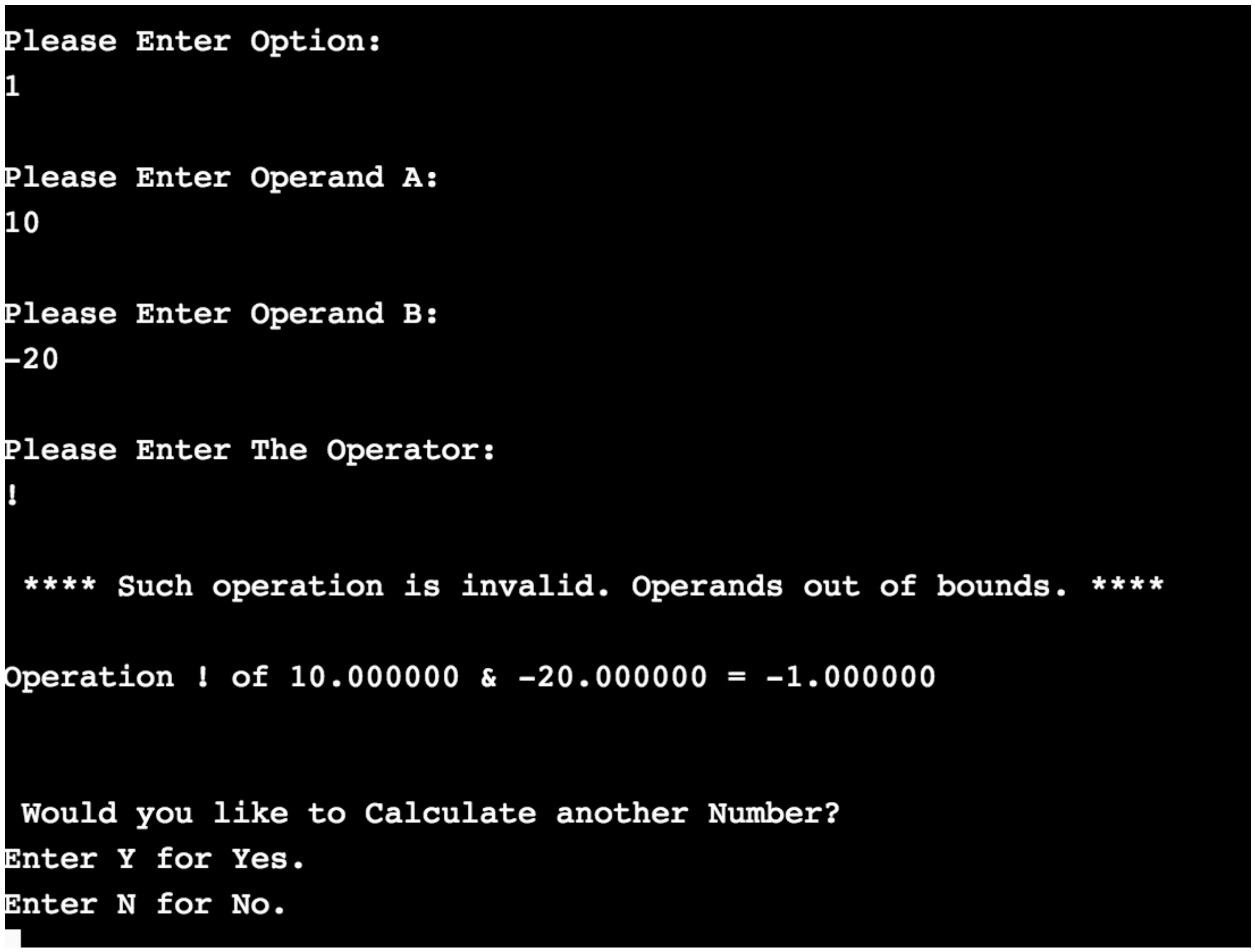
```
Please Enter Option:
1

Please Enter Operand A:
8

Please Enter Operand B:
2

Please Enter The Operator:
!

Operation ! of 8.000000 & 2.000000 = 3628800.000000


 Would you like to Calculate another Number?
Enter Y for Yes.
Enter N for No.
```

- Entered option 1.
- Entered 8 for operand A.
- Entered 2 for operand B.
- Entered '!' for the factorial operator
- Correct output answer is 3,628,800.   Since the sum of 8+2 = 10 and the !10 is 3,628,800.
- Entered 'Y' for yes.



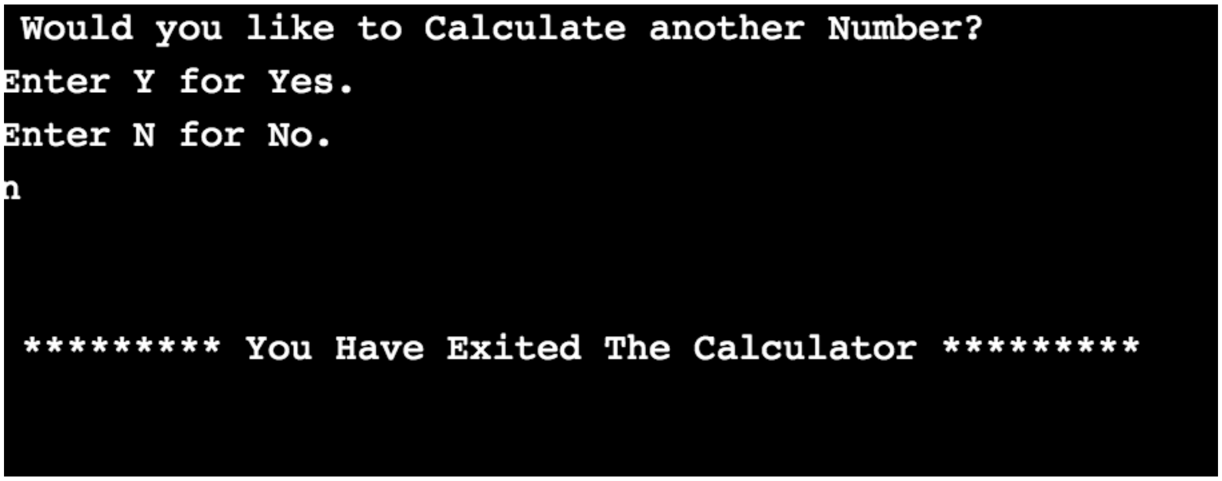
```
Please Enter Option:
1

Please Enter Operand A:
-5

Please Enter Operand B:
5

Please Enter The Operator:
!

Operation ! of -5.000000 & 5.000000 = 1.000000


 Would you like to Calculate another Number?
Enter Y for Yes.
Enter N for No.
```

- Entered option 1.
- Entered -5 as operand A.
- Entered 5 as operand B.
- Entered ‘!’ for the factorial operator.
- Correct answer output is 1, since the factorial of any sum of 0 (!0) equals 1.
- Entered ‘Y’ for yes.



```
Please Enter Option:
1

Please Enter Operand A:
10

Please Enter Operand B:
-20

Please Enter The Operator:
!

 **** Such operation is invalid. Operands out of bounds. ****

Operation ! of 10.000000 & -20.000000 = -1.000000


 Would you like to Calculate another Number?
Enter Y for Yes.
Enter N for No.
```

- Entered option 1.
- Entered 10 for operand A.
- Entered -20 for operand B.
- Entered '!' for the factorial operator.
- Error Message displayed since you cannot create a factorial from a negative sum of numbers. -1 is displayed as an output for error.
- Entered 'N' for no.



```
 Would you like to Calculate another Number?
Enter Y for Yes.
Enter N for No.
n


 ********* You Have Exited The Calculator *********
```

- Exit message to show that the calculator program has ended.

Program Output

```
****************************

Welcome to the Calculator

****************************

This Calulator will Complete the following operations between two operands:
 + (Addition)
 - (Subtraction)
 X (Multiplication)
 P (Exponent Power)
 ! (Factorial)

Note that the Subtraction operator subtracts Operand A from Operand B.
 Example:   B - A = C
 Note that the decimal exponent of a power will be rounded
 Example: 1^3.33 will be calculated as 1^3

Note that the Factorial operator will find the factorial of the sum of the two operands rounded
to the nearest whole number.
 Example: 3.4+2=5  !5= 5x4x3x2x1 = 120.

All Numbers that are round, are rounded by one decimal place.

        **** Menu ****


Option 1: Perform operation with new operands.

Option 2: Perform an operation with Answer from previous operation.

Please Enter 1 for Option 1. Please Enter 2 For Option 2.
*****************************************************************

Please Enter Option:
1

Please Enter Operand A:
5
```

```
Please Enter Operand B:
5□ □5.33

Please Enter The Operator:
+

Operation + of 5.000000 & 5.330000 = 10.330000


 Would you like to Calculate another Number?
Enter Y for Yes.
Enter N for No.
Y

This Calulator will Complete the following operations between two operands:
 + (Addition)
 - (Subtraction)
 X (Multiplication)
 P (Exponent Power)
 ! (Factorial)

Note that the Subtraction operator subtracts Operand A from Operand B.
 Example:   B - A = C
 Note that the decimal exponent of a power will be rounded
 Example: 1^3.33 will be calculated as 1^3

Note that the Factorial operator will find the factorial of the sum of the two operands rounded
to the nearest whole number.
 Example: 3.4+2=5  !5= 5x4x3x2x1 = 120.

All Numbers that are round, are rounded by one decimal place.

        **** Menu ****


Option 1: Perform operation with new operands.

Option 2: Perform an operation with Answer from previous operation.

Please Enter 1 for Option 1. Please Enter 2 For Option 2.
****************************************************************

Please Enter Option:
```

```
2

Please Enter Operand A:
5

The Previous Answer will be used for Operand B.

Please Enter The Operator:
-

Operation - of 5.000000 & 10.330000 = 5.330000


 Would you like to Calculate another Number?
Enter Y for Yes.
Enter N for No.
Y

This Calulator will Complete the following operations between two operands:
 + (Addition)
 - (Subtraction)
 X (Multiplication)
 P (Exponent Power)
 ! (Factorial)

Note that the Subtraction operator subtracts Operand A from Operand B.
 Example:   B - A = C
 Note that the decimal exponent of a power will be rounded
 Example: 1^3.33 will be calculated as 1^3

Note that the Factorial operator will find the factorial of the sum of the two operands rounded
to the nearest whole number.
 Example: 3.4+2=5  !5= 5x4x3x2x1 = 120.

All Numbers that are round, are rounded by one decimal place.

        **** Menu ****


Option 1: Perform operation with new operands.

Option 2: Perform an operation with Answer from previous operation.

Please Enter 1 for Option 1. Please Enter 2 For Option 2.
```

```
*****************************************************************

Please Enter Option:
2

Please Enter Operand A:
6

The Previous Answer will be used for Operand B.

Please Enter The Operator:
X

Operation X of 6.000000 & 5.330000 = 31.980000


 Would you like to Calculate another Number?
Enter Y for Yes.
Enter N for No.
y

This Calulator will Complete the following operations between two operands:
 + (Addition)
 - (Subtraction)
 X (Multiplication)
 P (Exponent Power)
 ! (Factorial)

Note that the Subtraction operator subtracts Operand A from Operand B.
 Example:   B - A = C
 Note that the decimal exponent of a power will be rounded
 Example: 1^3.33 will be calculated as 1^3

Note that the Factorial operator will find the factorial of the sum of the two operands rounded
to the nearest whole number.
 Example: 3.4+2=5  !5= 5x4x3x2x1 = 120.

All Numbers that are round, are rounded by one decimal place.

        **** Menu ****


Option 1: Perform operation with new operands.
Option 2: Perform an operation with Answer from previous operation.

Please Enter 1 for Option 1. Please Enter 2 For Option 2.
****************************************************************

Please Enter Option:
2

Please Enter Operand A:
2

The Previous Answer will be used for Operand B.

Please Enter The Operator:
P

Operation P of 2.000000 & 31.980000 = 4294967296.000000


 Would you like to Calculate another Number?
Enter Y for Yes.
Enter N for No.
y

This Calulator will Complete the following operations between two operands:
 + (Addition)
 - (Subtraction)
 X (Multiplication)
 P (Exponent Power)
 ! (Factorial)

Note that the Subtraction operator subtracts Operand A from Operand B.
 Example:   B - A = C
 Note that the decimal exponent of a power will be rounded
 Example: 1^3.33 will be calculated as 1^3

Note that the Factorial operator will find the factorial of the sum of the two operands rounded
to the nearest whole number.
 Example: 3.4+2=5  !5= 5x4x3x2x1 = 120.

All Numbers that are round, are rounded by one decimal place.

        **** Menu ****
```

```
Option 1: Perform operation with new operands.

Option 2: Perform an operation with Answer from previous operation.

Please Enter 1 for Option 1. Please Enter 2 For Option 2.
*****************************************************************

Please Enter Option:
1

Please Enter Operand A:
32⍰ ⍰⍰ ⍰4

Please Enter Operand B:
6.55

Please Enter The Operator:
p

Operation p of 4.000000 & 6.550000 = 16384.000000


 Would you like to Calculate another Number?
Enter Y for Yes.
Enter N for No.
Y


****************************

Welcome to the Calculator

****************************

This Calulator will Complete the following operations between two operands:
 + (Addition)
 - (Subtraction)
 X (Multiplication)
 P (Exponent Power)
 ! (Factorial)

Note that the Subtraction operator subtracts Operand A from Operand B.
 Example:   B - A = C
 Note that the decimal exponent of a power will be rounded
 Example: 1^3.33 will be calculated as 1^3

Note that the Factorial operator will find the factorial of the sum of the two operands rounded
to the nearest whole number.
 Example: 3.4+2=5  !5= 5x4x3x2x1 = 120.

All Numbers that are round, are rounded by one decimal place.

        **** Menu ****


Option 1: Perform operation with new operands.

Option 2: Perform an operation with Answer from previous operation.

Please Enter 1 for Option 1. Please Enter 2 For Option 2.
****************************************************************

Please Enter Option:
1

Please Enter Operand A:
-6

Please Enter Operand B:
-8.2

Please Enter The Operator:
p

 **** Out of Bounds *****

Operation p of -6.000000 & -8.200000 = -1.000000


 Would you like to Calculate another Number?
Enter Y for Yes.
Enter N for No.
y

This Calulator will Complete the following operations between two operands:
 + (Addition)
 - (Subtraction)
 X (Multiplication)
 P (Exponent Power)
 ! (Factorial)

Note that the Subtraction operator subtracts Operand A from Operand B.
 Example:   B - A = C
 Note that the decimal exponent of a power will be rounded
 Example: 1^3.33 will be calculated as 1^3

Note that the Factorial operator will find the factorial of the sum of the two operands rounded
to the nearest whole number.
 Example: 3.4+2=5  !5= 5x4x3x2x1 = 120.

All Numbers that are round, are rounded by one decimal place.

        **** Menu ****


Option 1: Perform operation with new operands.

Option 2: Perform an operation with Answer from previous operation.

Please Enter 1 for Option 1. Please Enter 2 For Option 2.
*******************************************************************

Please Enter Option:
1

Please Enter Operand A:
-6

Please Enter Operand B:
4� �5

Please Enter The Operator:
p

Operation p of -6.000000 & 5.000000 = -7776.000000


 Would you like to Calculate another Number?
Enter Y for Yes.
Enter N for No.
y
```

```
This Calulator will Complete the following operations between two operands:
 + (Addition)
 - (Subtraction)
 X (Multiplication)
 P (Exponent Power)
 ! (Factorial)

Note that the Subtraction operator subtracts Operand A from Operand B.
 Example:   B - A = C
 Note that the decimal exponent of a power will be rounded
 Example: 1^3.33 will be calculated as 1^3

Note that the Factorial operator will find the factorial of the sum of the two operands rounded
to the nearest whole number.
 Example: 3.4+2=5  !5= 5x4x3x2x1 = 120.

All Numbers that are round, are rounded by one decimal place.

        **** Menu ****


Option 1: Perform operation with new operands.

Option 2: Perform an operation with Answer from previous operation.

Please Enter 1 for Option 1. Please Enter 2 For Option 2.
*******************************************************************

Please Enter Option:
1

Please Enter Operand A:
3

Please Enter Operand B:
-6.44

Please Enter The Operator:
p

Operation p of 3.000000 & -6.440000 = 0.001372
```

```
 Would you like to Calculate another Number?
Enter Y for Yes.
Enter N for No.
y

This Calulator will Complete the following operations between two operands:
 + (Addition)
 - (Subtraction)
 X (Multiplication)
 P (Exponent Power)
 ! (Factorial)

Note that the Subtraction operator subtracts Operand A from Operand B.
 Example:   B - A = C
 Note that the decimal exponent of a power will be rounded
 Example: 1^3.33 will be calculated as 1^3

Note that the Factorial operator will find the factorial of the sum of the two operands rounded
to the nearest whole number.
 Example: 3.4+2=5  !5= 5x4x3x2x1 = 120.

All Numbers that are round, are rounded by one decimal place.

        **** Menu ****


Option 1: Perform operation with new operands.

Option 2: Perform an operation with Answer from previous operation.

Please Enter 1 for Option 1. Please Enter 2 For Option 2.
***************************************************************

Please Enter Option:
1

Please Enter Operand A:
0

Please Enter Operand B:
3

Please Enter The Operator:
p
```

```
Operation p of 0.000000 & 3.000000 = 0.000000


 Would you like to Calculate another Number?
Enter Y for Yes.
Enter N for No.
y

This Calulator will Complete the following operations between two operands:
 + (Addition)
 - (Subtraction)
 X (Multiplication)
 P (Exponent Power)
 ! (Factorial)

Note that the Subtraction operator subtracts Operand A from Operand B.
 Example:   B - A = C
 Note that the decimal exponent of a power will be rounded
 Example: 1^3.33 will be calculated as 1^3

Note that the Factorial operator will find the factorial of the sum of the two operands rounded
to the nearest whole number.
 Example: 3.4+2=5  !5= 5x4x3x2x1 = 120.

All Numbers that are round, are rounded by one decimal place.

        **** Menu ****


Option 1: Perform operation with new operands.

Option 2: Perform an operation with Answer from previous operation.

Please Enter 1 for Option 1. Please Enter 2 For Option 2.
*****************************************************************

Please Enter Option:
1

Please Enter Operand A:
-2

Please Enter Operand B:
```

```
1

Please Enter The Operator:
p

Operation p of -2.000000 & 1.000000 = -2.000000


 Would you like to Calculate another Number?
Enter Y for Yes.
Enter N for No.
y

This Calulator will Complete the following operations between two operands:
 + (Addition)
 - (Subtraction)
 X (Multiplication)
 P (Exponent Power)
 ! (Factorial)

Note that the Subtraction operator subtracts Operand A from Operand B.
 Example:   B - A = C
 Note that the decimal exponent of a power will be rounded
 Example: 1^3.33 will be calculated as 1^3

Note that the Factorial operator will find the factorial of the sum of the two operands rounded
to the nearest whole number.
 Example: 3.4+2=5  !5= 5x4x3x2x1 = 120.

All Numbers that are round, are rounded by one decimal place.

        **** Menu ****


Option 1: Perform operation with new operands.

Option 2: Perform an operation with Answer from previous operation.

Please Enter 1 for Option 1. Please Enter 2 For Option 2.
*****************************************************************

Please Enter Option:
1
```

```
Please Enter Operand A:
6543

Please Enter Operand B:
0

Please Enter The Operator:
p

Operation p of 6543.000000 & 0.000000 = 1.000000


 Would you like to Calculate another Number?
Enter Y for Yes.
Enter N for No.
y

This Calulator will Complete the following operations between two operands:
 + (Addition)
 - (Subtraction)
 X (Multiplication)
 P (Exponent Power)
 ! (Factorial)

Note that the Subtraction operator subtracts Operand A from Operand B.
 Example:   B - A = C
 Note that the decimal exponent of a power will be rounded
 Example: 1^3.33 will be calculated as 1^3

Note that the Factorial operator will find the factorial of the sum of the two operands rounded
to the nearest whole number.
 Example: 3.4+2=5  !5= 5x4x3x2x1 = 120.

All Numbers that are round, are rounded by one decimal place.

        **** Menu ****


Option 1: Perform operation with new operands.

Option 2: Perform an operation with Answer from previous operation.

Please Enter 1 for Option 1. Please Enter 2 For Option 2.
*****************************************************************
```

```
Please Enter Option:
1

Please Enter Operand A:
8

Please Enter Operand B:
2

Please Enter The Operator:
!

Operation ! of 8.000000 & 2.000000 = 3628800.000000


 Would you like to Calculate another Number?
Enter Y for Yes.
Enter N for No.
y

This Calulator will Complete the following operations between two operands:
 + (Addition)
 - (Subtraction)
 X (Multiplication)
 P (Exponent Power)
 ! (Factorial)

Note that the Subtraction operator subtracts Operand A from Operand B.
 Example:   B - A = C
 Note that the decimal exponent of a power will be rounded
 Example: 1^3.33 will be calculated as 1^3

Note that the Factorial operator will find the factorial of the sum of the two operands rounded
to the nearest whole number.
 Example: 3.4+2=5  !5= 5x4x3x2x1 = 120.

All Numbers that are round, are rounded by one decimal place.

        **** Menu ****


Option 1: Perform operation with new operands.
Option 2: Perform an operation with Answer from previous operation.

Please Enter 1 for Option 1. Please Enter 2 For Option 2.
*****************************************************************

Please Enter Option:
1

Please Enter Operand A:
-5

Please Enter Operand B:
5

Please Enter The Operator:
!

Operation ! of -5.000000 & 5.000000 = 1.000000


 Would you like to Calculate another Number?
Enter Y for Yes.
Enter N for No.
y

This Calulator will Complete the following operations between two operands:
 + (Addition)
 - (Subtraction)
 X (Multiplication)
 P (Exponent Power)
 ! (Factorial)

Note that the Subtraction operator subtracts Operand A from Operand B.
 Example:   B - A = C
 Note that the decimal exponent of a power will be rounded
 Example: 1^3.33 will be calculated as 1^3

Note that the Factorial operator will find the factorial of the sum of the two operands rounded
to the nearest whole number.
 Example: 3.4+2=5  !5= 5x4x3x2x1 = 120.

All Numbers that are round, are rounded by one decimal place.

        **** Menu ****
```

```
Option 1: Perform operation with new operands.

Option 2: Perform an operation with Answer from previous operation.

Please Enter 1 for Option 1. Please Enter 2 For Option 2.
****************************************************************

Please Enter Option:
1

Please Enter Operand A:
10

Please Enter Operand B:
-20

Please Enter The Operator:
!

 **** Such operation is invalid. Operands out of bounds. ****

Operation ! of 10.000000 & -20.000000 = -1.000000


 Would you like to Calculate another Number?
Enter Y for Yes.
Enter N for No.
n


 ********* You Have Exited The Calculator *********
```