

## 1. Odd or Even

### Aim:

To check whether a give integer is odd or even.

### Algorithm:

Step 1: Start

Step 2: Read the integer input by the user.

Step 3: Check if the number is divisible by 2.

Step 4: If yes, print that it is even.

Step 5: If not, print that it is odd.

Step 6: End

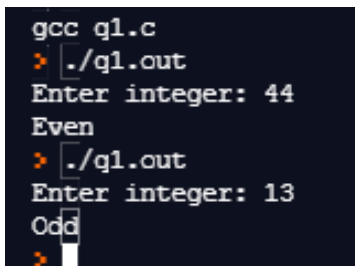
### Code:

```
//to check whether a given integer is odd or even
#include <stdio.h>

int main(void) {
    int num;
    printf("Enter integer: ");
    scanf("%d",&num);
    if (num%2==0)
        printf("Even\n");
    else
        printf("Odd\n");

    return 0;
}
```

### Output:



```
gcc q1.c
> ./q1.out
Enter integer: 44
Even
> ./q1.out
Enter integer: 13
Odd
>
```

### Result:

A program for determining whether a given integer is odd or even is written and executed.

## 2. Temperature Conversion

### Aim:

To convert a temperature given in Celsius to Fahrenheit and Kelvin.

### Algorithm:

Step 1: Start

Step 2: Read the temperature in Celsius as input from the user.

Step 3: Convert the temperature to Fahrenheit using the formula  $F = (9/5)C + 32$  and to Kelvin using the formula  $K = C + 273.15$ .

Step 4: Print the temperatures in the Fahrenheit and Kelvin scales.

Step 5: End

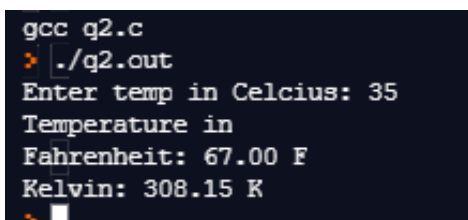
### Code:

```
//to convert the given temp in Celsius to Fahrenheit and Kelvin scale
#include <stdio.h>

int main(void) {
    float c,f,k;
    printf("Enter temp in Celsius: ");
    scanf("%f",&c);
    f=(9/5)*c+32;
    k=c+273.15;
    printf("Temperature in\nFahrenheit: %.2f F\nKelvin: %.2f K\n", f,k);

    return 0;
}
```

### Output:



```
gcc q2.c
> ./q2.out
Enter temp in Celcius: 35
Temperature in
Fahrenheit: 67.00 F
Kelvin: 308.15 K
>
```

### Result:

A program for converting a given temperature from Celsius to Fahrenheit and Kelvin is written and executed.

### 3. Identifying if Odd or Even

**Aim:**

To identify a give integer as odd or even using the conditional operator.

**Algorithm:**

Step 1: Start

Step 2: Read the integer as input from the user. Also, read a flag variable.

Step 3: Use the conditional operator to check if the integer is divisible by 2 – if yes, set flag as 1 and if no, set flag as 0.

Step 4: Print the value of the flag variable.

Step 5: End

**Code:**

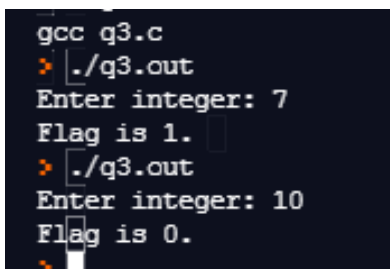
```
//to change the value of a flag depending on whether a given integer is odd or even
#include <stdio.h>

int main(void) {
    int num, flag=1;
    printf("Enter integer: ");
    scanf("%d",&num);

    flag=((num%2==0)?1:0); //conditional operator

    printf("Flag is %d.\n",flag);

    return 0;
}
```

**Output:**

```
gcc q3.c
./q3.out
Enter integer: 7
Flag is 1.
./q3.out
Enter integer: 10
Flag is 0.
```

**Result:**

A program for identifying a given integer as odd or even is written and executed.

## 4. Net Pay

### Aim:

To find the net pay of an employee given the basic pay as input.

### Algorithm:

Step 1: Start

Step 2: Read the basic pay (BP) as input from the user.

Step 3: Calculate the HRA (88% BP), DA (8% BP), PF (10% BP) from the BP based on the given formulae.

Step 4: Set the CCA and insurance to be 1000 and 2000 respectively.

Step 5: Calculate the gross pay as the sum of BP, DA, HRA and CCA; the net deduction as the sum of insurance and PF.

Step 6: Store the net pay as the difference of the gross pay and net deduction.

Step 7: Print the net pay.

Step 8: End

### Code:

```
//to find the net pay of an employee given the basic pay as input
#include <stdio.h>

int main(){
    float bp, da, hra, cca, ins, pf, gp, ded, np;
    //input
    printf("Enter value of Basic Pay: ");
    scanf("%f", &bp);
    //calculation
    da=0.88*bp;
    hra=0.08*bp;
    cca=1000;
    ins=2000;
    pf=0.1*bp;
    gp=bp+da+hra+cca;
    ded=ins+pf;
    np=gp-ded;
    //result
    printf("Net pay: %.1f\n", np);

    return 0;
}
```

**Output:**

```
gcc q4.c
./q4.out
Enter value of Basic Pay: 1000
Net pay: 860.0
```

**Result:**

A program to calculate the net pay of an employee given the basic pay is written and executed.

## 5. Conditional Net Pay

### Aim:

To find the net pay of an employee given the basic pay and subject to some categorical conditions.

### Algorithm:

Step 1: Start

Step 2: Read the basic pay (BP) as input from the user.

Step 3: Read the city type and designation of the employee.

Step 4: Calculate the HRA and CCA based on the given formulae and the appropriate category as given in the question. The other values are calculated as done in the previous case.

Step 5: Calculate the gross pay as the sum of BP, DA, HRA and CCA; the net deduction as the sum of insurance and PF.

Step 6: Store the net pay as the difference of the gross pay and net deduction.

Step 7: Print the net pay.

Step 8: End

### Code:

```
//to find the net pay of an employee given the basic pay as input
#include <stdio.h>
```

```
int main(){
    float bp, da, hra, cca, ins, pf, gp, ded, np;
    char city,desgn;

    printf("Enter value of Basic Pay: ");
    scanf("%f",&bp);

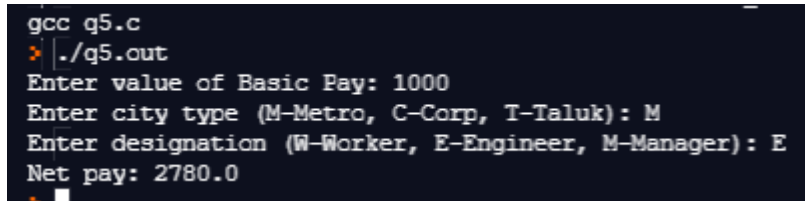
    printf("Enter city type (M-Metro, C-Corp, T-Taluk): ");
    scanf(" %c",&city);
    printf("Enter designation (W-Worker, E-Engineer, M-Manager): ");
    scanf(" %c",&desgn);

    da=0.88*bp;
    //hra
    switch(city){
        case 'M': hra=0.1*bp; break;
        case 'C': hra=0.08*bp; break;
        case 'T': hra=0.05*bp; break;
        default: printf("Invalid city type");
    }
    //cca
    switch(desgn){
        case 'W': cca=1000; break;
```

```
    case 'E': cca=2000; break;
    case 'M': cca=5000; break;
    default: printf("Invalid designation");
}
ins=2000;
pf=0.1*bp;
gp=bp+da+hra+cca;
ded=ins+pf;
np=gp-ded;

printf("Net pay: %.1f\n", np);

return 0;
}
```

**Output:**

```
gcc q5.c
./q5.out
Enter value of Basic Pay: 1000
Enter city type (M-Metro, C-Corp, T-Taluk): M
Enter designation (W-Worker, E-Engineer, M-Manager): E
Net pay: 2780.0
```

**Result:**

A program for computing the conditional net pay is written and executed.

## 6. Pattern

### Aim:

To print the given pattern as is, getting the reference number as input.

### Algorithm:

Step 1: Start

Step 2: Read any integer N between 3 and 10 as input from the user. Validate the input.

Step 3: For the upper half of the pattern, run a loop with i running from N to 1. Go to step 7 when i=0.

Step 4: Print whitespaces until the position is reached where the \*'s will be centred.

Step 5: Fill up the centres with \*'s or \*- 's accordingly to get the pattern.

Step 6: Go to a new line after the completion of the line and go to step 3 with the decremented value of i.

Step 7: For the lower half of the pattern, run a loop with i running from 2 to N. Go to step 10 when i=N+1.

Step 8: Fill the line with whitespaces and \*'s in the appropriate positions to complete the pattern.

Step 9: Go to a new line after the completion of the line and go to step 7 with the incremented value of i.

Step 10: End

### Code:

```
//to print the given pattern
#include<stdio.h>

int main(){
    int N, flag=0;
    while(flag==0){
        printf("Enter any number between 3 and 10: ");
        scanf("%d", &N);
        if (N>0)
            {flag=1; break;}
    }

    printf("\n");
    //upper half
    for (int i=N; i>=1; i--)
    {
        for (int k=0; k<=N-i; k++)
        {
            printf(" ");
        }
        for (int j=1; j<=i; j++)
        {
            if (j==i){ printf("*"); }
            else { printf("*-");}
        }
        printf("\n");
    }
```



```

}
//lower half
for (int i=2; i<=N; i++)
{
    for (int k=0; k<= N-i; k++)
    {
        printf(" ");
    }
    for (int j=1; j<=i; j++)
    {
        if (j==i){ printf("*"); }
        else { printf("*-");}
    }
    printf("\n");
}

return 0;
}

```

### Output:

```
gcc q6.c
./q6.out
Enter any number between 3 and 10: 5

*_*_*_*_*
 *_*_*_*
  *_*_*
   *_*
    *
   *_*
  *_*_*
 *_*_*_*
*_*_*_*_*
```

**Result:**

A program to print the given pattern is written and executed.

## 7. Simple Interest

### Version 1

#### Aim:

To find the sum of any given N integers.

#### Algorithm:

Step 1: Start

Step 2: Read a number N as input from the user.

Step 3: Introduce a sum and set its value to be 0.

Step 4: Run a loop N number of times.

Step 5: For each iteration, read an integer as input from the user and add it to the sum.

Step 6: Exit the loop after N iterations and print the sum.

Step 7: End

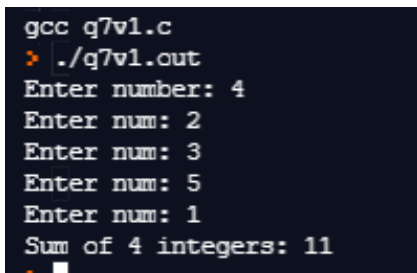
#### Code:

```
//to find the sum of any given N integers
#include <stdio.h>

int main(void) {
    int N,num,sum=0;
    printf("Enter number: ");
    scanf("%d",&N);
    for (int i=1; i<=N; i++){
        printf("Enter num: ");
        scanf("%d",&num);
        sum+=num;
    }
    printf("Sum of %d integers: %d\n", N,sum);

    return 0;
}
```

#### Output:



```
gcc q7v1.c
> ./q7v1.out
Enter number: 4
Enter num: 2
Enter num: 3
Enter num: 5
Enter num: 1
Sum of 4 integers: 11
```

**Result:**

A program (version 1) to calculate the sum of the give list of integers is written and executed.

**Version 2****Aim:**

To find the sum of any given N integers for any value of N and terminating when N=-999 is entered.

**Algorithm:**

Step 1: Start

Step 2: Read a number N as input from the user.

Step 3: Enter a loop to keep getting the value of N as input from the user until the user enters -999 for N.

Step 4: In every loop, introduce a sum and set its value to be 0.

Step 5: Run a loop N number of times.

Step 6: For each iteration, read an integer as input from the user and add it to the sum.

Step 7: Exit the integer loop after N iterations and print the sum.

Step 8: Get the next value of N and repeat steps 4-7 until N=-999, when the loop is terminated.

Step 7: End

**Code:**

```
//to find the sum of N integers for many N's until N=-999
```

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

```
int main(void) {
    int N,num,sum;
    //input
    printf("Enter number: ");
    scanf("%d",&N);
    do{
        sum=0;
        //calculation
        for (int i=1; i<=N; i++){
            printf("Enter num: ");
            scanf("%d",&num);
            sum+=num;
        }
        //output
        printf("Sum of given %d integers: %d\n", N,sum);
        //new input
        printf("Enter number: ");
        scanf("%d",&N);
    }while(N!=-999);
}
```

```
    return 0;  
}
```

**Output:**

```
gcc q7v2.c  
> ./q7v2.out  
Enter number: 6  
Enter num: 1  
Enter num: 3  
Enter num: 0  
Enter num: -2  
Enter num: -5  
Enter num: 10  
Sum of given 6 integers: 7  
Enter number: 2  
Enter num: 23  
Enter num: 51  
Sum of given 2 integers: 74  
Enter number: -999  
>
```

**Result:**

A program (version 2) to calculate the sum of the give list of integers is written and executed.

**Version 3****Aim:**

To find the sum of any given N integers until "STOP" is entered.

**Algorithm:**

Step 1: Start

Step 2: Read a number N as input from the user.

Step 3: Enter a loop to keep getting the value of N as input from the user until the user enters "STOP".

Step 4: In every loop, introduce a sum and set its value to be 0.

Step 5: Run a loop N number of times.

Step 6: For each iteration, read an integer as input from the user and add it to the sum.

Step 7: Exit the integer loop after N iterations and print the sum.

Step 8: Get the next value of N and repeat steps 4-7 until N="STOP", when the loop is terminated.

Step 7: End

**Code:**

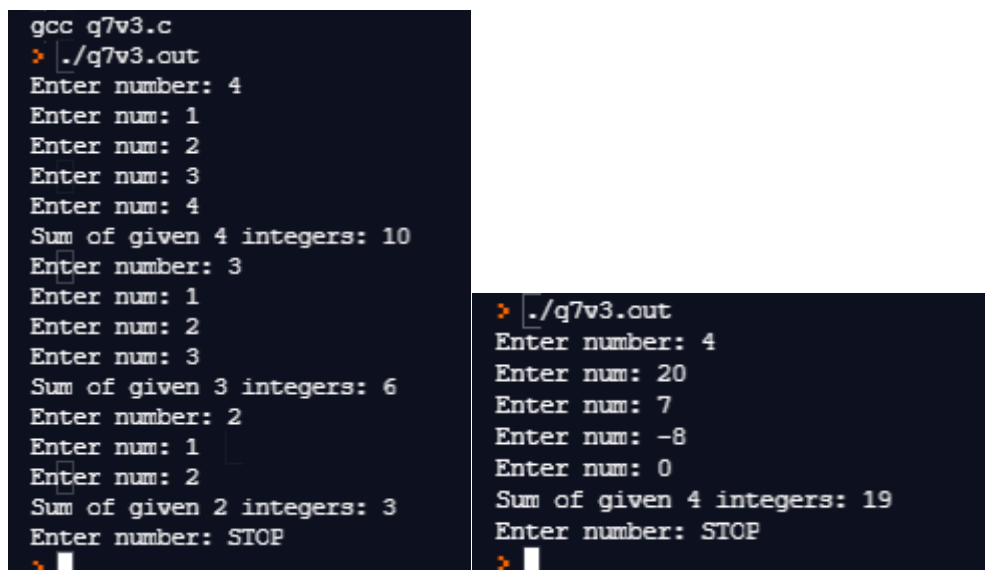
```
//to find the sum of N integers for many N's until STOP is entered.  
#include <stdio.h>
```

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

int main(void) {
    int N,num,sum;
    char a[10];
    //input
    printf("Enter number: ");
    scanf("%s",a);
    do{
        sum=0;
        N=atoi(a);
        //calculation
        for (int i=1; i<=N; i++){
            printf("Enter num: ");
            scanf("%d",&num);
            sum+=num;
        }
        //output
        printf("Sum of given %d integers: %d\n", N,sum);
        //new input
        printf("Enter number: ");
        scanf("%s",a);
    }while(strcmp(a,"STOP")!=0);

    return 0;
}
```

### Output:



```
gcc q7v3.c
> ./q7v3.out
Enter number: 4
Enter num: 1
Enter num: 2
Enter num: 3
Enter num: 4
Sum of given 4 integers: 10
Enter number: 3
Enter num: 1
Enter num: 2
Enter num: 3
Sum of given 3 integers: 6
Enter number: 2
Enter num: 1
Enter num: 2
Sum of given 2 integers: 3
Enter number: STOP
>

> ./q7v3.out
Enter number: 4
Enter num: 20
Enter num: 7
Enter num: -8
Enter num: 0
Sum of given 4 integers: 19
Enter number: STOP
>
```

### Result:

A program (version 3) to calculate the sum of the give list of integers is written and executed.

Version 4**Aim:**

To find the sum of any given N integers validating N to be a positive integer less than 100.

**Algorithm:**

Step 1: Start

Step 2: Read a number N as input from the user.

Step 3: Enter a loop to keep getting the value of N as input from the user until the user enters -999 for N.

Step 4: In every loop, introduce a sum and set its value to be 0.

Step 5: Check if N is a positive integer less than 100. If yes, go to step 6. If no, get the next value of N.

Step 6: Run a loop N number of times.

Step 7: For each iteration, read an integer as input from the user and add it to the sum.

Step 8: Exit the integer loop after N iterations and print the sum.

Step 9: Get the next value of N and repeat steps 4-8 until N=-999, when the loop is terminated.

Step 10: End

**Code:**

```
//to find the sum of N integers when 0<N<100
#include <stdio.h>

int main(void) {
    int N,num,sum;
    //input
    printf("Enter a positive number less than 100: ");
    scanf("%d",&N);

    while(N!=-999){
        sum=0;
        if (N>0 && N<100){
            //calculation
            for (int i=1; i<=N; i++){
                printf("Enter num: ");
                scanf("%d",&num);
                sum+=num;
            }
            //output
            printf("Sum of given %d integers: %d\n", N,sum);
        }
        //new input
        printf("Enter a positive number less than 100: ");
        scanf("%d",&N);
    }
}
```

```
    return 0;  
}
```

**Output:**

```
gcc q7v4.c  
➤ ./q7v4.out  
Enter a positive number less than 100: 5  
Enter num: 12  
Enter num: 42  
Enter num: -20  
Enter num: 8  
Enter num: 10  
Sum of given 5 integers: 52  
Enter a positive number less than 100: 2  
Enter num: 18  
Enter num: -2  
Sum of given 2 integers: 16  
Enter a positive number less than 100: 102  
Enter a positive number less than 100: 0  
Enter a positive number less than 100: -4  
Enter a positive number less than 100: 3  
Enter num: 1  
Enter num: 2  
Enter num: 3  
Sum of given 3 integers: 6  
Enter a positive number less than 100: -999  
➤
```

**Result:**

A program (version 4) to calculate the sum of the give list of integers is written and executed.

**Version 5****Aim:**

To find the sum of any given N integers validating N to be a positive integer less than 100 and if invalid, to exit after displaying an error message.

**Algorithm:**

Step 1: Start

Step 2: Read a number N as input from the user.

Step 3: Enter a loop to keep getting the value of N as input from the user until the user enters -999 for N.

Step 4: In every loop, introduce a sum and set its value to be 0.

Step 5: Check if N is a positive integer less than 100. If yes, go to step 6. If no, display an error message and go to step 10.

Step 6: Run a loop N number of times.

Step 7: For each iteration, read an integer as input from the user and add it to the sum.

Step 8: Exit the integer loop after N iterations and print the sum.

Step 9: Get the next value of N and repeat steps 4-8 until N=-999, when the loop is terminated.

Step 10: End

### Code:

```
//to find the sum of N integers when 0<N<100 and display an error message for invalid input, then exit
```

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

```
int main(void) {
    int N,num,sum;
    //input
    printf("Enter a positive number less than 100: ");
    scanf("%d",&N);

    while(N!=-999){
        sum=0;
        if (N>0 && N<100){
            //calculation
            for (int i=1; i<=N; i++){
                printf("Enter num: ");
                scanf("%d",&num);
                sum+=num;
            }
            //output
            printf("Sum of given %d integers: %d\n", N,sum);
        }
        else{
            printf("Invalid input! Exiting...\n");
            break;
        }
        //new input
        printf("Enter a positive number less than 100: ");
        scanf("%d",&N);
    }

    return 0;
}
```



**Output:**

```
gcc q7v5.c
./q7v5.out
Enter a positive number less than 100: 4
Enter num: 1
Enter num: 2
Enter num: 3
Enter num: -2
Sum of given 4 integers: 4
Enter a positive number less than 100: -1
Invalid input! Exiting...
```

**Result:**

A program (version 5) to calculate the sum of the give list of integers is written and executed.

**Version 6****Aim:**

To find the sum of any given N integers validating N to be a positive integer less than 100 and if invalid, to prompt for entering again after displaying an error message.

**Algorithm:**

Step 1: Start

Step 2: Read a number N as input from the user.

Step 3: Enter a loop to keep getting the value of N as input from the user until the user enters -999 for N.

Step 4: In every loop, introduce a sum and set its value to be 0.

Step 5: Check if N is a positive integer less than 100. If yes, go to step 6. If no, display an error message and prompt the user to re-enter the value of N.

Step 6: Run a loop N number of times.

Step 7: For each iteration, read an integer as input from the user and add it to the sum.

Step 8: Exit the integer loop after N iterations and print the sum.

Step 9: Get the next value of N and repeat steps 4-8 until N=-999, when the loop is terminated.

Step 10: End

**Code:**

```
//to find the sum of N integers when 0<N<100 and display an error message for invalid
input, then ask for another input
#include <stdio.h>
```

```
int main(void) {
    int N,num,sum;
    //input
    printf("Enter a positive number less than 100: ");
    scanf("%d",&N);

    while(N!=-999){
        sum=0;
        if (N>0 && N<100){
            //calculation
            for (int i=1; i<=N; i++){
                printf("Enter num: ");
                scanf("%d",&num);
                sum+=num;
            }
            //output
            printf("Sum of given %d integers: %d\n", N,sum);
        }
        else{
            printf("Invalid input! Please enter a different value.\n");
        }
        //new input
        printf("Enter a positive number less than 100: ");
        scanf("%d",&N);
    }

    return 0;
}
```

**Output:**

```
> gcc q7v6.c
> ./q7v6.out
Enter a positive number less than 100: 4
Enter num: 1
Enter num: 2
Enter num: 3
Enter num: -1
Sum of given 4 integers: 5
Enter a positive number less than 100: 2
Enter num: 12
Enter num: 20
Sum of given 2 integers: 32
Enter a positive number less than 100: -40
Invalid input! Please enter a different value.
Enter a positive number less than 100: 102
Invalid input! Please enter a different value.
Enter a positive number less than 100: 6
Enter num: 12
Enter num: 10
Enter num: -30
Enter num: -4
Enter num: 23
Enter num: -5
Sum of given 6 integers: 6
Enter a positive number less than 100: -999
>
```

**Result:**

A program (version 6) to calculate the sum of the give list of integers is written and executed.

## 8. Calculator

### Aim:

To design a basic calculator with addition, subtraction, multiplication, division and squaring functions.

### Algorithm:

Step 1: Start

Step 2: Read two numbers as input from the user.

Step 3: Provide the user with options for addition, subtraction, multiplication, division and squaring and store the choice.

Step 4: If the choice is addition, add the two numbers and store the result.

Step 5: Else, if the choice is subtraction, subtract the second number from the first and store the result.

Step 6: Else, if the choice is multiplication, multiply the two numbers and store the result.

Step 7: Else, if the choice is division, divide the first number by the second and store the result.

Step 8: Else, if the choice is squaring, ask the user which number to square, square that number and store the result.

Step 9: Print the result.

Step 10: End

### Code:

```
//to design a calculator
#include<stdio.h>
#include<math.h>

int main(void){
    float a,b,n,res;
    //input values
    printf("Input two numbers: ");
    scanf("%f %f",&a,&b);

    int ch;
    printf("Options:-
\n1:Addition\n2:Subtraction\n3:Multiplication\n4:Division\n5:Squaring\n");
    scanf("%d",&ch);

    switch(ch){
        case 1: res=a+b; break;
        case 2: res=a-b; break;
        case 3: res=a*b; break;
        case 4: res=a/b; break;
        case 5: {
            printf("Which number do you wish to square? %f or %f?: ", a,b);
            scanf("%f",&n);
```

Date: 31/05/2021

```
res=pow(n,2);
break;
}
default: printf("Invalid choice");
}

printf("Result: %.1f\n",res);

return 0;
}
```

**Output:**

```
gcc q8.c -lm
> ./q8.out
Input two numbers: 12 4
Options:-
1:Addition
2:Subtraction
3:Multiplication
4:Division
5:Squaring
1
Result: 16.0
> ./q8.out
Input two numbers: 12 4
Options:-
1:Addition
2:Subtraction
3:Multiplication
4:Division
5:Squaring
4
Result: 3.0
> ./q8.out
Input two numbers: 12 4
Options:-
1:Addition
2:Subtraction
3:Multiplication
4:Division
5:Squaring
5
Which number do you wish to square? 12.000000 or 4.000000?: 4
Result: 16.0
>
```

**Result:**

A program to implement a basic calculator is written and executed.

## 9. Consecutive 5's

### Aim:

To check if a number contains three consecutive 5's or not.

### Algorithm:

Step 1: Start

Step 2: Read a number as input from the user. Also, read a counter and set it to 0.

Step 3: Run a loop as long as the number is greater than 0 and check the rightmost digit of the number. If the number is not greater than 0, terminate the loop and go to step 7.

Step 4: If the digit is 5, increment the counter. If not, set the counter to 0.

Step 5: If the counter reaches 3, exit the loop.

Step 6: Divide the number by 10, store the result as the new value of the number and go to step 3.

Step 7: If the counter is 3, print "YES". Otherwise, print "NO".

Step 8: End

### Code:

```
//to check if a number has 3 consecutive 5s or not
```

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

```
int main(void) {
    int num,ctr=0;
    printf("Number: ");
    scanf("%d",&num);

    while (num>0){
        if (num%10==5){
            ctr++;
            if (ctr==3)
                break;
        }
        else
            ctr=0;
        num/=10;
    }
    if (ctr>=3)
        printf("Result: YES\n");
    else
        printf("Result: NO\n");

    return 0;
}
```

**Output:**

```
gcc q9.c
> ./q9.out
Number: 375554
Result: YES
>
> ./q9.out
Number: 375155
Result: NO
> ./q9.out
Number: 763552
Result: NO
>
```

**Result:**

A program to check if a given number contains three consecutive 5's or not is written and executed.

## 10. Odd or Even (Alt. method)

### Aim:

To check whether a given integer is odd or even without using conditionals.

### Algorithm:

Step 1: Start

Step 2: Read an integer as input from the user.

Step 3: Create an array with the index 0 corresponding to "Even" and index 1 corresponding to "Odd".

Step 4: Print the value in the array corresponding to the index which is the remainder of the number when divided by 2.

Step 5: End

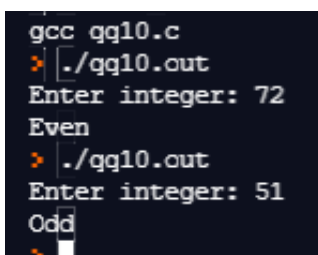
### Code:

```
//to check whether a given integer is odd or even without using conditionals.
```

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

```
int main(void) {  
    int num;  
    printf("Enter integer: ");  
    scanf("%d",&num);  
  
    char* a[2]={"Even","Odd"};  
    printf("%s\n",a[num%2]);  
  
    return 0;  
}
```

### Output:



```
gcc qq10.c  
./qq10.out  
Enter integer: 72  
Even  
./qq10.out  
Enter integer: 51  
Odd
```

### Result:

A program to check whether a given integer is odd or even is written and executed.