Question No. 1

Title: WAP in c/c++ to find the ceiling and floor value of a given input float number.

Theory:

Ceiling value of a number is the smallest integer that is greater than or equal to the given number. It rounds the number upwards to the nearest integer. Ceiling value of number is represented by $\lceil 8.2 \rceil$ and value of $\lceil 8.2 \rceil = 9$.

Floor value of a number is the largest integer that is less than or equal to the given number. It rounds downwards to the nearest integer. floor value of number 8.2 is represented as $\lfloor 8.2 \rfloor$ and its equivalent value is 8.

Procedure

1. Input declaration: • Declare a variable 'x' for input of floating number to stored.
   • Declare suitable name for variable (int) for storing results.
2. Input prompt: prompt the user to enter input (float) number.
3. Take Input: using cin take the number (float) as input.
4. Calculate floor value and ceiling value using floor() and ceiling function ceil().
5. Display Result: output the floor value and ceiling value.
6. Terminate the program: End the program.

Source code:

```
#include<iostream>
#include<math.h>
  using namespace std;
```

```
int main()
{
    float r, floor_value, ceiling_value;
    cout<<"Enter any float number for its ceiling value
        and ceiling value :"<<endl;
    cin>>r;
    floor_value = floor(r);
    ceiling_value = ceil(r);
    cout<<" The floor value of the number is "<<floor_value
        <<endl;
    cout<<" The ceiling value of the number is "<<ceiling-
      value<<endl;
      return 0;
}
```

floor

output:

Enter the floating point number for its ~~ceiling~~ floor value and ceiling value.

9.99

The floor value of the number is 9.

The floor value of the number is 10.

conclusion:

The floor value and ceiling value of 9.99 was computed as 9 and 10 respectively by using a floor() and ceil() function in c++.

Question No-2

Title: WAP in c/c++ to find the cartesian product of two sets.

Theory:

Cartesian product: The cartesian product of two sets A and B, denoted by A×B is the set of all possible ordered pairs where the first element comes from set A and the second elements comes from set B.

$$A \times B = \{(a, b) \mid a \in A \text{ and } b \in B\}$$

procedure:

① Declare variables for number of elements in set A and B (n and m). Take input list prompt the user to enter no of elements.

② Declare two array A[n] and B[m] for two sets and take inputs for elements.

③ use two loops to compute and display the cartesian product A×B as ordered pairs.

④ ensure proper formatting for the output of the cartesian product(e.g. including commas between pairs).

⑤ End the program with 'return 0'.

source code:

```
#include<iostream>
using namespace std;
int main()
{
  int n,m,i,j;
  cout<<" enter the number of elements of set A."
     <<endl;
  cin>>n;
```

```
int A[n];
cout<<"Enter the elements of set A."<<endl;
for(i=0;i<n;i++)
{
  cin>>A[i];
}
cout<<"Enter the number of elements of set B."<<
      endl;
cin>>m;
int B[m];
cout<<"Enter the elements of set B simultaneously"<<endl;
for(j=0;j<m;j++)
{
  cin>>B[j];
}
cout<<"The cartesian product of set A and set B is:"
   <<endl<<"AxB = {";
for(i=0;i<n;i++)
{
  for(j=0;j<n;j++)
  {
  ~~cout<<"(A[i],"<<B[j]<<~~
  cout<<"("<<A[i]<<","<<B[j]<<")";
  if(i!=n-1||j!=m-1)
  {
  cout<<",";
  }
}
}
cout<<"}"<<endl;
return 0;
}
```

Output:

```
Enter the number of elements of set A
4
Enter the elements of set A simultaneously.
1    2    3    4
Enter the number of elements of set B
2
Enter the number of elements of set B simultaneously.
4    5
The cartesian product of set A and set B is:
AXB = {(1,4), (1,5), (2,4), (2,5), (3,4), (3,5), (4,4),
       (4,5)}
```

Conclusion:

The cartesian product of two sets AXB was computed using the c++ programming Language.

Question no-3

Title: WAP in C/C++ to find the permutation by asking the value of n and r from user.

Theory:

permutation: A mathematical calculation of number of ways a particular set can be arranged. permutation is the arrangement of items in which order matters. Mathematically,

$$^{n}P_{r} = P(n,r) = \frac{n!}{(n-r)!} \quad \text{where } n \geq r$$

permutation are useful in various field such as cryptography, scheduling problems and game theory and many fields.

Procedure / Algorithm

1) Declaration of n, r, for input and npr for result.
2) Input values: Read integers n and r from user.
3) Calculate n!: compute factorial of n.
4) Calculate (n-1)! = compute factorial of n-1
5) Compute permutation: use the formula $npr = P(n,r) = \frac{n!}{(n-r)!}$
6) Display result in output screen.
7) End program: Terminate the program with return 0 successfully.

Source code:

```
#include<iostream>
  using namespace std;
  int main()
{
  int n, r, fact_of_n=1, fact_of_n_minus_r=1, npr;
```

```
cout<<" enter the value of n and r to compute the
  permutation "<<endl;
cin>>n>>r;
for(int i=1;i<n;i++)
{
  fact_of_n=fact_of_n*i;
}
 for(int i=1 ; i<n-r; i++)
{
  fact-of-n-minus-r= fact_of_n-minus_r*i;
}
 npr= fact_of_n / fact_of_n-minus_r;
 cout<<" permutation npr is = "<<npr<<endl;
  return 0;
}
```

Output

Enter the value of n and r to compute the permutation

5

2

The permutation npr is = 20.

Conclusion: The permutation $5P_2$ was calculated as 20. by using c++ programming language

Question No-4

Title: WAP in c/c++ to find the combination by asking value of n and r from user.

Theory:

Combination is a selection of items from a larger set where the order of the items does not matter unlike permutation, combinations focus solely on choosing elements without regard to the arrangement.

The number of combinations of selecting r elements from a set of n elements is given by the formula.

$$C(n,r) = \frac{n!}{r!\,(n-r)!}$$

The application of combinations in computer science field are cryptography, Database query optimization, machine learning, Algorithm Design, Data Compression and many more.

Procedure/Algorithm

1) Input Declaration: Define variables for 'n', 'r' and factorial value
2) Input values: Get value of 'n' and 'r' from user.
3) Compute factorial: use code and logic to compute factorial n!, r! and (n-r)! using loops.
4) Apply combination formula: compute nCr = n! (r! x (n-r)!)
5) Display the resut: output the value of combination.
6) Find the program: Return '0' and exist the program successfully.

Source code:

```
#include <iostream>
using namespace std;
```

```
int main()
#include <iostream>
{
  int i,n,r,j=1,k=1,l=1,ncr;
  cout<<" enter the value of n and r to compute combi-
  nation \n";
  cin>>n>>r;
  for(i=1;i<=n;i++)
  {
    j=j*i;
  }
  for(i=1;i<=n-r;i++)
  {
    k=k*i;
  }
  for(i=1;i<=r;i++)
  {
  l=l*i;
  }
  ncr=j/(k*l);
  cout<<"The combination nCr="<<ncr<<endl;
  return 0;
}
```

output:

Enter the value of n and r to compute combination

7

4

The combination nCr=35.

Conclusion: The combination $^7C_4$ was calculated as 35 by using c++ programming language

Question No-5

Title: WAP in c/c++ to find the factorial of an input number using recursive function.

Theory:

Recursion is a method where function calls itself to solve a smaller instance of the same problem. It involves a base case and called termination case to terminates the recursion and a recursive case to continue breaking down the problem. Recursion concept is used in various mathematical computation, Data structures, Divide and conquer algorithm, etc.

Factorial:

Factorial is the product of positive integers less than or equal to a given number n. It is represented by n! where 0! is defined as 1.

Procedure/Algorithm

1. Start by reading the input number n.
2. Call the recursive function fact(n).
3. In fact() function, if n=0 or 1, return 1 (base case).
4. Otherwise, return n*fact(n-1) (recursive case)
5. print the computed factorial and end the program.

Recursive Definition for factorial of non-negative integer.

$$\boxed{fn = n \times fn-1}$$ where, $n = 0$ i.e. $fn = 1$

Source code:

```
#include <iostream>
using namespace std;
int fact(int n)
{
 if(n==0 || n==1)
```

```
{
  return 0;
}
else
{
  return n* fact(n-1);
}
}
int main()
{
  int n, factorial;
  cout<<" Enter a number for the computation of the
  factorial "<<endl;
  cin>>n;
  factorial = fact(n);
  cout<<" The factorial of the number "<<n<<" is "
  << factorial << endl;
  return 0;
}
```

~~Output:~~

```
Enter a number for the computation of the factorial
6
The factorial of the number 6 is 720.
```

Conclusion:

By using c++ program, the factorial of an input number '6' is completed as '720' using the recursive function.

Question No-6

Title: WAP in C/C++ to find the nth term of a fibonacci sequence using recursive function.

Theory: Recursion is a method where function calls itself to solve a smaller instance of same problem. It involves a base case which terminates the recursion and a recursive case to continue the program by breaking down the problem.

Application of Recursion: Mathematical computation, Data structure, Divide and conquer algorithm etc

Fibonacci sequence:

A sequence $\{a_n\}$ of $\{f_n\}$ is said as fibonacci sequence of $a_0=0$, $a_1=1$ $a_n = a_{n-1} + a_{n+2}$ for $n \geq 2$.

$\{a_n\} = 0, 1, 1, 2, 3, 5, 8, 13 \ldots$

Procedure/Algorithm:

1. Start by reading the input number 'n'
2. Define a recursive function fib(n) to calculate the nth fibonacci number.
3. In the fib(n) function, if n=0 return 0, and if n=1 return 1 (base case)
4. otherwise, return fib(n-1)+ fib (n-2) (recursive case).
5. Call fib(n) to compute the fibonacci number.
6. print the result.
7. End the program.

Source code:

```
#include <iostream>
using namespace std;
int fib(n)
{
   if (n=0)
```

```
{
  return 0;
}
  else if(n==1 || n==2)
  {
   return (0+1);
  }
  else
  {
   return  (fib(n-1)+fib(n-2));
  }
}
 int main()
{
 int n;
 cout<<" Enter the value of nth term of the fibonacci
     number"<<endl;
 for(int i=0; i<=n; i++)
 {
  cout<<" fib(i);
 if(i!=n) { cout<<","<<" "; }
 cout<<"\n\n The "<<n<<" th term of sequence is
    fib(n)<<endl;
 return 0;
}
```

output:

```
Enter the value of n in nth term of the fibonacci number
9
The fibonacci sequence up to nth term is:
0, 1, 1, 2, 3, 5, 8, 13, 21, 34
The 9th term of the sequence is 34
```

Conclusion:

Hence, By using c++ programming language the nth term of the fibonacci sequence was computed. The 9th term of fibonacci sequence was 34.

Question No-7

Title: WAP in c/c++ to raise the base of an input number by a certain power using recursive function.

Theory:

Recursion is a method where function calls itself to solve a smaller instance of same problem. It involves a base case which terminates the recursion and a recursive case to continue the program by down the problem.

Application of Recursion:

Mathematical computation, Data structure, Divide and conquer algorithm etc

b raise to power n:

The mathematical expression "b raised to the power n" denoted by b^n means multiplying the base b by itself n times. formally

$$b^n = b \times b \times b \ldots \times b \text{ (n times)}$$

For eg:-

$$10^3 = 10 \times 10 \times 10 = 1000$$

Algorithm and procedure:

1. Start the program.
2. Input the base 'b' and exponent 'n'.
3. Check if n==0: if true, return 1, (base case)
4. Recursive step: otherwise call power (b, n-1) and multiply the result b i.e. b*power (b, n-1).
5. Return the result of the multiplication as the result of the current recursive call.
6. Display the final result.
7. End the program.

Source code:

```
#include<iostream>
using namespace std;
int power(int b, int n)
{
  if(n==0)
  {
    return 1;
  }
  else
  {
    return b*power(b,n-1);
  }
}
int main()
{
  int b, n, result;
  cout<<"enter the value of base"<<endl;
  cin>>b;
  cout<<"enter the value of exponent"<<endl;
  cin>>n;
  result=power(b,n);
  cout<<"b raise to power n is "<<result<<endl;
  return 0;
}
```

Output:

```
enter the value of base
7
enter the value of exponent
4
b raise to power n is 2401.
```

Conclusion:

Hence, by using c++ programming language we computed the b raise to power n by the concept of recursive function and the value of 7 raise to the power 4 is 2401.

Question No:- 8

Title: WAP in c/c++ to perform linear search by using recursive function.

Theory:

Linear search is an searching algorithm used to find the position of the target element (key) in an array. It works by checking each element of array sequentially starting from the 1st element and moving towards the last, until the key is found or the entire array has been searched.

Linear search is good for small amount of data sets in array. As each element need to be checked large data (array) is not preferred to search by linear search.

Procedure/Algorithm

1. Start the program.
2. Read the number n for n no. of elements.
3. Read 'n' no. of elements in array.
4. Read the key which is to be searched.
5. call the function linear-search (A, 0, n-1, key) to initiate the search.
6. Base case: Linear_search() if the key matches A[low], print the success ~~method~~ message with index low.
7. Recursive case: If the key does not match A[low], recursively call linear_search (A, low+1, high, key) to search the next element.
8. The recursion stops if key found and print result in function.
9. Terminate the program

Source code:

```
#include<iostream>
using namespace std;
void linear_search(int A[], int low, int key)
int main()
{
  int n, i, key;
  cout<<" enter the no. of elements of array "<<endl;
  cin>>n;
  int A[n];
  cout<<" enter the "<<n<<" elements of the array"<<
    endl;
  for(i=0; i<n; i++)
  {
   cin>>A[i];
  }
  cout<<"\n enter the value of data (integer) data you
   want to search "<<endl;
  cin>>key;
  linear_search(A, 0, n-1, key);
  return 0;
}
 void linear_search(int A[], int low, int high, int key)
{
   if(key== A[low])
  {
  cout<<" searching successful in location "<<low<<endl;
  }
  else
{
  linear_search(A, low+1, high, key);
```

3

3

output:

```
Enter the no. of elements of array
5
Enter the 5 elements of the array
8   4   23   12   7
Enter the value of data (integer) you want to search.
12
Searching successful in location 3.
```

conclusion:

Hence, by using c++ programming language we computed linear search by using recursive function. And the searching successful location is in 3 was computed.

Question No:- 9

Title: WAP in c/c++ to perform binary search by using recursive function.

Theory:

Binary search is an efficient algorithm for finding an item from a sorted array of elements. It works by repeatedly dividing the search interval in half.

1) Compare the target value with the middle element of the sorted array.
2) Decide the next step:
   (a) If the target value equals the middle element the search is successful.
   (b) If the target is less, search the left half, if greater, search the right half.
3) Repeat or End: continue the search on the chosen half until the target is found or the interval is empty.

Procedure/Algorithm

1) Start the program
2) Initialize the array in ascend order of elements for binary search.
3) Compute mid-point and call the function binary_search (A, low, high, key)
4) check key: If key == A[mid], print success with mid position.
   - if key < A[mid], search the left half: binary_search (A, low, mid-1, key)
   - if key > A[mid], search the right half: binary_search (A, mid+1, high, key)
5) Terminate if failure of searching
6) Terminate the program.

Source code

```
#include <iostream>
using namespace std;
void binary_search (int A[], int low, int high, int key)
{
 int mid= (high+low)/2;
 if (key== A[mid])
 {
  cout<<"search successful at location"<<mid<<endl;
}
else if (key< A[mid])
{
  binary_search (A, low, mid-1, key);
}
else if (key>A[mid])
{
  binary_search (A, mid+1, high, key);
}
 else
{
  cout<<"unsuccessful to search the key"<<endl;
}
int main()
{
  int [A]= {1, 2, 3, 4, 6, 7, 8, 13, 193
  int key, low=0, high=19;
cout<<"enter the data you want to search:"<<endl;
cin>>key;
binary_search (A, low, high, key);
return 0;
}
```

Output:

```
Enter the data you want to search
4
Search successful at location 3.
```

Conclusion:

In conclusion, the program successfully implements the Binary Search algorithm to find the largest value (key) within an array.

Question No:- 10

Title: WAP in c/c++ to implement Euclidean Algorithm for computing GCD.

Theory:

The GCD (Greatest common Divisor) to two integers that divides both numbers without leaving a rimined for example, the GCD of 24 and 36 is 12, as the 12 is the largest number that divides both 24 and 36 exactly.

Application:

Fraction simplification, cryptography, LCM calculation, Modular Arithmatic etc

Euclidean Algorithm is an efficient method to compute the GCD of two numbers. It works by repeatedly replasing the larger number by its remainder when divided by the smaller number until one of the numbers becomes zero.

Euclidean Algorithm for computing gcd of a and b

1. start
2. read 'a' and 'b' from the user.
3. while b>0, do

   3.1: set r=a mod b;

   3.2: a=b

   3.3: b=r
4. return 'a' as gcd
5. print the gcd of a and b
6. Terminate the program.

Source code:

```
#include<iostream>
using namespace std;
int main()
{
 int a, b, r;
 cout<<" enter two non-zero positive integer for
  gcd" <<endl;
  cin>>a>>b;
 cout<<" The HCD of "<<a<<" and "<<b<<" is";
 while (b>0)
 {
  r= a%b;
  a= b;
  b=r;
}
 cout<<"a<< endl;
 return 0;
}
```

Output:

```
Entir two non-zero positive integers for gcd.
12
30
The HCD of 12 and 30 is 6.
```

Conclusion:

By using c++ programming language, the gcd of two integers wis computed by the Euclidean Algorithm. The gcd of 120 and 30 was found as 6.

Question No:- 11

Title: WAP in C/C++ to implement extended Euclidean Algorithm for computing GCD.

Theory:

The Extended Euclidean Algorithm is an extension of Euclidean algorithm, which is used to compute the greatest common divisor (gcd) of two integers. In addition to finding GCD the extended Euclidean algorithm expresses the gcd as a linear combination of the two integers, a and b it finds integers x and y such that:

$$GCD(a,b) = ax + by$$

Procedure / Algorithm:

Input: Two non-negative integers a and b $a \geq b$.
output: $d = gcd(a,b)$ and integers, x and y satisfying $ax+by=d$.

1. If $b=0$ then set $d=a$, $x=1$, $r=0$, and return d(x,y)
2. Set $x_2=1$, $x_1=0$, $y_2=0$, $y_1=1$
3. while $(b>0)$, do

   a). $q = floor(a/b)$, $r=a-qb$, $x=x_2-qx_1$, $y=y_2-qy_1$

   b) $a=b$, $b=r$, $x_2=x_1$, $x_1=x$, $y_2=y$, $y_1=y$.
4. Set $d=a$, $x=x_2$, $y=y_2$ and return d(x,y).
5. Terminate the program.

Source code:

```
#include <iostream>
#include <Math.h>
 using namespace std;
 int main()
{
 int a, b, q, r, d, x=1, y=0;
```

```
cout<<" Enter the two numbers a and b for their
GCD (a≥b and they must be non negative "<<endl;
cin>>a>>b;
int p=a, s=b;
if (b==0)
{
  d=a;
cout<<" GCD of "<<a<<" and "<<b<<" is "<<d
      <<endl;
cout<<d"<<" = "<<a<<". "<<a<<"+"<<b<<"."
<<y<<endl;
return 0;
}
cout<<"GCD of "<<a<<" and "<<b<<" is ";
int x1=0, x2=1, y1=1, y2=0;
while (b>0)
{
 q= floor(a/b);
 r= a- q*b;
 x = x2 - q*x1;
y = y2 - q x y1;
 a=b;
 b=r;
 x2=x1;
 x1=x;
 y1= y;
}
 d=a;
 x=x1;
y=y2;
cout<<d<<endl;
```

```
cout << "Linear combination is:";
cout << d << "=" << p << "." << x << "+" << s << "." << y << endl;
return 0;
}
```

output:

```
Enter the two numbers a and b for their GCD
(a>=b and they must be positive)
10
12
GCD of 10 and 12 is 2
Linear combination is 2 = 10.1 + 12.-1
```

conclusion:

By using c++ programming language, the implementation of Extended Euclidean Algorithm was performed for computing the GCD of two numbers. GCD of 10 and 12 was computed as 2.