C++ Assignment

1. W.A.P to count the digits in a given supplied integer number.

e.g: if given integer is int data= 12345. output:- digit=5

Ans-

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

int main() {
int data=12345, count =0;
for(; data > 0; data /= 10)
count++;
cout<<"Digits = " <<count<<endl;
return 0;
}
```

2. W.A.P to calculate the sum of all digits in a given supplied integer number.

Input: int data=1234

Output : sum of all digits is 10

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

int main() {
   int data = 1234;
   int sum = 0;
   for (; data != 0; data /= 10)
      sum = sum + data % 10;
   cout << "Sum of all digits is " << sum << endl;
   return 0;
}
```

3. W.A.P to calculate the sum of all even digits in a given supplied integer number.
Input: int data=1234
Output : sum of all digits is 6
here 2,4 are the even digit.
Ans-

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

int main() {
   int data = 1234, reversed = 0, sumEven = 0;
   while (data > 0) {
      int digit = data % 10;
      reversed = reversed * 10 + digit;
      data /= 10;
   }
   data = reversed;
   cout << "Even digits: ";
   while (data > 0) {
      int digit = data % 10;
      if (digit % 2 == 0) {
         sumEven = sumEven + digit;
         cout << digit;
         if (data / 10 != 0) {
            cout << ",";
         }
      }
      data /= 10;
   }
   cout << endl << "Sum of even digits: " << sumEven << endl;
   return 0;
}
```

4. W.A.P to calculate the sum of all odd digits in a given supplied integer number.
Input: int data=1234
Output : sum of all digits is 4
here 1,3 are the odd digit.

Ans-

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

int main() {
   int data = 1234, reversed, sumOdd = 0;
   while (data > 0) {
      int digit = data % 10;
      reversed = reversed * 10 + digit;
      data /= 10;
   }
   data = reversed;
   cout << "Odd digits: ";
   while (data > 0) {
      int digit = data % 10;
      if (digit % 2 != 0) {
         sumOdd += digit;
         cout << digit <<",";
      }
      data /= 10;
   }
   cout << endl << "Sum of odd digits: " << sumOdd << endl;
   return 0;
}
```

5. W.A.P to reverse the all digits in a given supplied integer number.
Input: int data=12345
output: data=54321.
Ans-

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

int main (){
int data = 12345;
cout << "Reversed number: ";
while (data > 0){
        int digit = data % 10;
        cout << digit;
        data /= 10;
        }
return 0;
}
```

6.W.A.P to delete the given digit in a given integer.
e.g: Input int data=132345;
digit=3;
Output: data=1245
Ans-

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

int removeDigit(int data, int digit) {
   int result = 0, place = 1;
   while (data > 0) {
      int currentDigit = data % 10;
      if (currentDigit != digit) {
         result = result + currentDigit * place;
         place *= 10;
      }
      data /= 10;
   }
   return result;
}
int main() {
   int data = 132345, digit = 3;
   cout << "Input data: " << data << endl;
   cout << "Digit to remove: " << digit << endl;
   cout << "Resulting data: " << removeDigit(data, digit) << endl;
   return 0;
}
```

7.W.A.P to check how many occurrences of given digit in a given integer number.
Case 1:
Intput : int data=12334.
int d=3;
Output: 3 have 2 occurrences
Case 2:
Intput : int data=12334.
int d=7;
Output: 7 have no any occurrences.

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

int main() {
   int number = 12334;
    cout << "Given Integer Number is: "<<number << endl;
   int count3 = 0, count7 = 0;
   while (number > 0) {
      int digit = number % 10;
      count3 += (digit == 3);
      count7 += (digit == 7);
      number /= 10;
   }
     if (count3 > 0) {
      cout << "Digit 3 have " << count3 << " occurrences." << endl;
   } else {
      cout << "Digit 3 have no any occurrences." << endl;
   }
   if (count7 > 0) {
      cout << "Digit 7 have " << count7 << " occurrences." << endl;
   } else {
      cout << "Digit 7 have no any occurrences." << endl;
   }
   return 0;
}
```

8.W.A.P to find occurrences of each digit in a given integer.
Case 1: e.g Input: int data=1277455 output: 1=0 2=0 7=1 4=0 5=1
Case 2: Input int data=11111 output: 1=4
Ans-

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

void countDigits(int data) {
   int count0 = 0, count1 = 0, count2 = 0, count3 = 0, count4 = 0;
   int count5 = 0, count6 = 0, count7 = 0, count8 = 0, count9 = 0;

   int tempData = data;
   while (data > 0) {
      int digit = data % 10;
      switch (digit) {
         case 0: count0++; break;
         case 1: count1++; break;
         case 2: count2++; break;
         case 3: count3++; break;
         case 4: count4++; break;
         case 5: count5++; break;
         case 6: count6++; break;
         case 7: count7++; break;
         case 8: count8++; break;
         case 9: count9++; break;
      }
      data /= 10;
   }

   if (count0 > 0) cout << "0=" << (count0 > 1 ? count0 - 1 : 0) << endl;
   if (count1 > 0) cout << "1=" << (count1 > 1 ? count1 - 1 : 0) << endl;
   if (count2 > 0) cout << "2=" << (count2 > 1 ? count2 - 1 : 0) << endl;
   if (count3 > 0) cout << "3=" << (count3 > 1 ? count3 - 1 : 0) << endl;
   if (count4 > 0) cout << "4=" << (count4 > 1 ? count4 - 1 : 0) << endl;
   if (count5 > 0) cout << "5=" << (count5 > 1 ? count5 - 1 : 0) << endl;
   if (count6 > 0) cout << "6=" << (count6 > 1 ? count6 - 1 : 0) << endl;
   if (count7 > 0) cout << "7=" << (count7 > 1 ? count7 - 1 : 0) << endl;
   if (count8 > 0) cout << "8=" << (count8 > 1 ? count8 - 1 : 0) << endl;
   if (count9 > 0) cout << "9=" << (count9 > 1 ? count9 - 1 : 0) << endl;
}
```

```
int main() {
    int data1 = 1277455;
    cout << "Case 1:" << endl;
    cout << "Input: " << data1 << endl;
    countDigits(data1);
    int data2 = 11111;
    cout << "\nCase 2:" << endl;
    cout << "Input: " << data2 << endl;
    countDigits(data2);

    return 0;
}
```

9. W.A.P to check digits in given integer number either Ascending or Descending Order.

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

bool isOrdered(int num, bool ascending) {
   int lastDigit = num % 10;
   num /= 10;

   while (num > 0) {
      int currentDigit = num % 10;
      if (ascending && currentDigit >= lastDigit) {
         return false;
      }
      if (!ascending && currentDigit <= lastDigit) {
         return false;
      }
      lastDigit = currentDigit;
      num /= 10;
   }
   return true;
}

int main() {
   int number;
   cout << "Enter an integer number: ";
   cin >> number;
   if (isOrdered(number, true)) {
      cout << "This is ascending order." << endl;
   } else if (isOrdered(number, false)) {
      cout << "This is descending order." << endl;
   } else {
      cout << "Given integer data not in order." << endl;
   }
   return 0;
}
```

10.W.A.P to sort (Ascending order)the digits in a given integer number,without using arrays ,strings.
e.g: Input int data=4263 output: 2346

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

int main() {
   int n, number[30];
   cout << "Enter the value of N: ";
   cin >> n;
   cout << "Enter the numbers: ";
   for (int i = 0; i < n; ++i)
      cin >> number[i];
   for (int i = 0; i < n - 1; ++i) {
      for (int j = i + 1; j < n; ++j) {
         if (number[i] > number[j]) {
            swap(number[i], number[j]);
         }
      }
   }
   cout << "Ascending Order : ";
   for (int i = 0; i < n; ++i)
   cout << number[i] << " ";
   return 0;
}
```

11.W.A.P to sort (Descending order)the digits in a given integer number,without using arrays ,strings.
e.g: Input int data=4263 output: 6432

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

int main() {
   int n, number[30];
   cout << "Enter the value of N: ";
   cin >> n;
   cout << "Enter the numbers: ";
   for (int i = 0; i < n; ++i)
   cin >> number[i];
   for (int i = 0; i < n - 1; ++i) {
      for (int j = i + 1; j < n; ++j) {
         if (number[i] < number[j]) {
            swap(number[i], number[j]);
         }
      }
   }
   cout << "Descending Order : ";
   for (int i = 0; i < n; ++i)
      cout << number[i] << " ";
   return 0;
}
```

12.W.A.P to find highest digit in a given supplied integer number.
e.g: Input int data=124455. Output: highest digit is 5

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

int main() {
   int data = 124455;
   int highestDigit = 0;
   while (data > 0) {
      int digit = data % 10;
      if (digit > highestDigit) {
         highestDigit = digit;
      }
      data /= 10;
   }
   cout << "The highest digit is " << highestDigit << endl;
   return 0;
}
```

13.W.A.P to find lowest digit in a given supplied integer number.
e.g: Input int data=114455.
Output: lowest digit is 1.

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

int main() {
   int data = 124455;
   int lowestDigit = 9;
   while (data > 0) {
      int digit = data % 10;
      if (digit < lowestDigit) {
         lowestDigit = digit;
      }
      data /= 10;
   }
   cout << "The lowest digit is " << lowestDigit << endl;
   return 0;
}
```

14.W.A.P to swap highest digit and lowest digit in a given supplied integer number.
e.g: Input int data=12345.Output: highest digit is 5 and lowest digit is 1
after swap data=52341

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

int main() {
   int data = 12345;
   int highestDigit = 0;
   int lowestDigit = 9;
   int temp = data;
   while (temp > 0) {
      int digit = temp % 10;
      if (digit > highestDigit) {
         highestDigit = digit;
      }
     if (digit < lowestDigit) {
         lowestDigit = digit;
      }
      temp /= 10;
   }
   cout << "Highest digit is: " << highestDigit << " and Lowest digit is: " << lowestDigit <<
endl;
   int swappedData = 0;
   temp = data;
   int place = 1;
   while (temp > 0) {
      int digit = temp % 10;
      if (digit == 5) {
         digit = 1;
      } else if (digit == 1) {
         digit = 5;
      }
      swappedData += digit * place;
      place *= 10;
      temp /= 10;
   }
   cout << "After swapping data: " << swappedData << endl;
   return 0;
}
```

15.W.A.P to find second highest digit in a given supplied integer number.
e.g: Input int data=124455.
Output: highest digit is 4.

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

int main() {
   int data = 124455;
   int highestDigit = 0;
   int secondHighestDigit = 0;
   int temp = data;
   while (temp > 0) {
      int digit = temp % 10;
      if (digit > highestDigit) {
         secondHighestDigit = highestDigit;
         highestDigit = digit;
      } else if (digit > secondHighestDigit && digit != highestDigit) {
         secondHighestDigit = digit;
      }
      temp /= 10;
   }
   cout << "Second highest digit is: " << secondHighestDigit << endl;
   return 0;
}
```

16.W.A.P to check given integer is prime number or not.

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

int main() {
  int i, n;
  bool is_prime = true;
  cout << "Enter a positive integer: ";
  cin >> n;
  if (n == 0 || n == 1) {
    is_prime = false;
  }
  for (i = 2; i <= n/2; ++i) {
    if (n % i == 0) {
      is_prime = false;
      break;
    }
  }
  if (is_prime)
    cout << n << " is a prime number";
  else
    cout << n << " is not a prime number";
  return 0;
}
```

17.W.A.P to print prime numbers in between given ranges

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

bool isPrime(int n){
   int count = 0;
   if(n < 2)
      return false;
   for(int i = 2;i < n/2; i++)
   {
      if(n % i == 0)
         return false;
   }
   return true;
}
int main()
{
   int lower, upper;
   lower=1,upper=10;
   for(int i = lower; i <= upper; i++)
      if(isPrime(i))
         cout << i << " ";
}
```

18. W.A.P to print first 10 prime numbers from given start number.
Explanation: suppose start is 30 so you have to print first 10 prime numbers from 30.

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

int main()
{
        int a, b, i, j, flag;
        cout << "Enter lower bound of the interval: ";
        cin >> a;
        cout << "Enter upper bound of the interval: ";
        cin >> b;
        cout << "Prime numbers between "
                << a << " and " << b << " are: ";
   for (i = a; i <= b; i++) {
                if (i == 1 || i == 0)
                        continue;
        flag = 1;
                for (j = 2; j <= i / 2; ++j) {
                        if (i % j == 0) {
                                flag = 0;
                                break;
                        }
                }
if (flag == 1)
                        cout << i << " ";
        }
        return 0;
}
```

19. W.A.P to print twins prime numbers in between given ranges.
Explanation:Suppose range is 1 to 50 so prime numbers between this range are
2,3,5,7,11,13,17,19,23,29,31,37,41,43,47
so twins prime numbers are(3,5),(11,13),(17,19),(29,31),(41,43)
difference between two prime number is only 2 hence they are twins prime number.

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

bool isPrime(int num) {
   if (num <= 1) return false;
   for (int i = 2; i <= sqrt(num); ++i) {
      if (num % i == 0) return false;
      }
      return true;
}
int main() {
   int start, end;
   cout << "Enter the range (start end): ";
   cin >> start >> end;
   cout << "Twin prime numbers between " << start << " and " << end << " are:" << endl;
   for (int num = start; num <= end - 2; ++num) {
      if (isPrime(num) && isPrime(num + 2)) {
         cout << "(" << num << "," << num + 2 << ") ";
      }
   }
   cout << endl;
   return 0;
}
```

20.W.A.P to check the given number is perfect number or not.
e.g: input int data=123
output: no
int data=6
output : yes
perfect number is a positive integer that is equal to the sum of its proper divisors. The smallest perfect number is 6, which is the sum of 1, 2, and 3.

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

bool isPerfect(int num) {
   if (num <= 1) return false;
   int sum = 1;
      for (int i = 2; i <= num / 2; ++i) {
      if (num % i == 0) {
         sum += i;
      }
   }
   return sum == num;
}
int main() {
   int num;
   cout << "Enter a positive integer: ";
   cin >> num;
   if (isPerfect(num)) {
      cout << num << " is a perfect number." << endl;
   } else {
      cout << num << " is not a perfect number." << endl;
   }
   return 0;
}
```

21.W.A.P to print the perfect numbers in between given range.

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

bool isPerfect(int num) {
    if (num <= 1) return false;
    int sum = 1;
    for (int i = 2; i <= num / 2; ++i) {
        if (num % i == 0) {
            sum += i;
        }
    }
    return sum == num;
}
int main() {
    int start, end;
    cout << "Enter the range (start end): ";
    cin >> start >> end;
    cout << "Perfect numbers between " << start << " and " << end << " are: ";
    for (int num = start; num <= end; ++num) {
        if (isPerfect(num)) {
            cout << num << " ";
        }
    }
    return 0;
}
```

22.W.A.P to check the given integer number is strong number or not.
a strong number whose sum of all digits' factorial is equal to the number 'n'.
e.g int data=145
output : yes. 145 is strong number (1! + 4! + 5! = 145)

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

int factorial(int n) {
   if (n == 0 || n == 1)
      return 1;
   int fact = 1;
   for (int i = 2; i <= n; ++i) {
      fact *= i;
   }
   return fact;
}
bool isStrongNumber(int num) {
   int originalNum = num;
   int sum = 0;
   while (num > 0) {
      int digit = num % 10;
      sum += factorial(digit);
      num /= 10;
      }
   return sum == originalNum;
}
int main() {
   int num;
   cout << "Enter a positive integer: ";
   cin >> num;
   if (isStrongNumber(num)) {
      cout << "Output: yes. " << num << " is strong number (" << num << " = ";
      int temp = num;
      bool first = true;
      while (temp > 0) {
         int digit = temp % 10;
         if (!first) {
            cout << " + ";
         }
         cout << digit << "!";
```

```
            first = false;
            temp /= 10;
        }
        cout << ")" << endl;
    } else {
        cout << "Output: no. " << num << " is not a strong number." << endl;
    }
    return 0;
}
```

23.W.A.P to print the factorial of each digit in given integer number.
Intput int data=123.
Output : 9 … (1! + 2! + 3! =9)

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

int factorial(int n) {
    if (n == 0 || n == 1)
        return 1;
    else
        return n * factorial(n - 1);
}
int main() {
    int number = 123;
    int digit;
    int sumFactorials = 0;
    int originalNumber = number;
    while (number > 0) {
        digit = number % 10;
        sumFactorials += factorial(digit);
        number /= 10;
    }
    cout << "Output: " << sumFactorials << " ... (" << originalNumber << " = ";
    number = originalNumber;
    bool first = true;
    while (number > 0) {
        digit = number % 10;
        if (!first) {
            cout << " + ";
        }
        cout << digit << "!";
        first = false;
        number /= 10;
    }
    cout << " = " << sumFactorials << ")";
    return 0;
}
```

24.W.A.P to check given integer number is Armstrong number or not
An Armstrong number is a number which equal to the sum of
the cubes of its individual digits. For example, 153 is an Armstrong
number as − 153 = (1)3 + (5)3 + (3)3
153 = 1 + 125 + 27 153 = 153.

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

int order(int x)
{
    int len = 0;
    while (x)
    {
        len++;
        x = x/10;
    }
    return len;
}
bool armstrong(int num, int len){
    int sum = 0, temp, digit;
    temp = num;
    while(temp != 0)
    {
        digit = temp % 10;
        sum = sum + pow(digit,len);
        temp /= 10;
    };
    return num == sum;
}
int main ()
{
    int num = 153, len;
    len = order(num);
    if (armstrong(num, len))
        cout << num << " is Armstrong";
    else
        cout << num << " is not Armstrong";
    return 0;
}
```

25. W.A.P to find next highest number of given integer number.
e.g: Input int data=123.
Output:data= 132(a next number is a number greater number and must be contains same digits)
intput int data=534.
Output : no next highest number for data.

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

void swap(char *a, char *b) {
    char temp = *a;
    *a = *b;
    *b = temp;
}
void findNext(char number[], int n) {
    int i;
    for (i = n-1; i > 0; i--) {
        if (number[i] > number[i-1]) {
            break;
        }
    }
    if (i == 0) {
        cout << "No next highest number is possible." << endl;
        return;
    }
    int x = number[i-1], smallest = i;
    for (int j = i+1; j < n; j++) {
        if (number[j] > x && number[j] < number[smallest]) {
            smallest = j;
        }
    }
    swap(&number[smallest], &number[i-1]);
```

```
   sort(number + i, number + n);
   cout << "Output: No next highest number for " << number;
}
int main() {
   char digits[] = "123";
   int n = strlen(digits);
  findNext(digits, n);
   return 0;
}
```

26. W.A.P to print the numbers in between given range whose sum of digit reduced and equal to 5(or you can supply any digit).

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

int sumOfDigits(int num) {
   int sum = 0;
   while (num > 0) {
      sum += num % 10;
      num /= 10;
      }
   return sum;
}
int main() {
   int start = 1;
   int end = 100;
   int targetSum = 5;
   cout << "Numbers between " << start << " & " << end << " whose sum of digits is
equals to " << targetSum << ": ";
   for (int i = start; i <= end; ++i) {
      if (sumOfDigits(i) == targetSum) {
         cout << i << " ";
      }
   }
   return 0;
}
```

27.W.A.P to print the numbers in between given range whose sum of digit reduced and equal to 5(or you can supply any digit) and that number must be in order(either ascending or descending).

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

int sumOfDigits(int num) {
   int sum = 0;
   while (num > 0) {
      sum += num % 10;
      num /= 10;
      }
   return sum;
}
int main() {
   int start = 1;
   int end = 100;
   int targetSum = 7;
   cout << "Numbers between " << start << " & " << end << " whose sum of digits is
equals to " << targetSum << ": ";
   for (int i = start; i <= end; ++i) {
      if (sumOfDigits(i) == targetSum) {
         cout << i << " ";
      }
   }
   return 0;
}
```

28. W.A.P to print the numbers in between given range whose sum of digit reduced and equal to 5(or you can supply any digit) and also that number must be contains at list 1 occurrence of supply digit.

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

bool containsDigit(int num, int digit) {
   string numStr = to_string(num);
   char digitChar = digit + '0';
   return numStr.find(digitChar) != string::npos;
}
int sumOfDigits(int num) {
   int sum = 0;
   while (num > 0) {
      sum += num % 10;
      num /= 10;
   }
   return sum;
}
int main() {
   int start = 1;
   int end = 100;
   int targetSum = 5;
   int requiredDigit = 3;
   cout << "Numbers between " << start << " and " << end << " whose sum of digits
equals " << targetSum
        << " and contain at least one " << requiredDigit << ":"<<'\n';
   for (int i = start; i <= end; ++i) {
      if (sumOfDigits(i) == targetSum && containsDigit(i, requiredDigit)) {
         cout << i <<" ";
      }
   }
   return 0;
}
```

29.W.A.P to check number is palindrome number or not.
Palindrome number is nothing but reverse is eqaul to that number.
e.g : input int data=151
output: YES.
Input int data=345
output : NO

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

bool isPalindrome(int number) {
   int originalNumber = number;
   int reversedNumber = 0;
   while (number > 0) {
      reversedNumber = reversedNumber * 10 + number % 10;
      number /= 10;
   }
```

```
    return originalNumber == reversedNumber;
}

int main() {
    int num1 = 121;
    int num2 = 345;
    if (isPalindrome(num1)) {
        cout << num1 << " : YES " << endl;
    } else {
        cout << num1 << " : NO " << endl;
    }
    if (isPalindrome(num2)) {
        cout << num2 << " : YES " << endl;
    } else {
        cout << num2 << " : NO " << endl;
    }
    return 0;
}
```

30. W.A.P to print the palindrome numbers in between given range.

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

bool isPalindrome(int number) {
    int originalNumber = number;
    int reversedNumber = 0;
    while (number > 0) {
        reversedNumber = reversedNumber * 10 + number % 10;
        number /= 10;
    }
    return originalNumber == reversedNumber;
}
int main() {
    int start = 1;
    int end = 100;
   cout << "Palindrome numbers between " << start << " and " << end << ": ";
    for (int i = start; i <= end; ++i) {
        if (isPalindrome(i)) {
            cout << i <<" ";}
 }
return 0;}
```