1)Find Frequency of Characters in a C-style String

****

```
#include <iostream>

using namespace std;
int main()
{
  char c[] = "C++ programming is not easy.", check = 'm';
  int count = 0;

  for(int i = 0; c[i] != '\0'; ++i)
  {
     if(check == c[i])
        ++count;
  }
  cout << "Frequency of " << check <<  " = " << count;
  return 0;
}
```

2)Find Frequency of Characters of a String Object

****

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

int main()
{
   string str = "C++ Programming is awesome";
   char checkCharacter = 'a';
   int count = 0;

   for (int i = 0; i < str.size(); i++)
   {
      if (str[i] ==  checkCharacter)
      {
         ++ count;
      }
   }

   cout << "Number of " << checkCharacter << " = " << count;

   return 0;
}
```

Example 1: Copy String Object

****

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

int main()
{
   string s1, s2;

   cout << "Enter string s1: ";
   getline (cin, s1);

   s2 = s1;

   cout << "s1 = "<< s1 << endl;
   cout << "s2 = "<< s2;

   return 0;
}
```

To copy c-strings in C++, strcpy() function is used.
Example 2. Copy C-Strings
****

```
#include <iostream>
#include <cstring>

using namespace std;

int main()
{
   char s1[100], s2[100];

   cout << "Enter string s1: ";
   cin.getline(s1, 100);

   strcpy(s2, s1);

   cout << "s1 = "<< s1 << endl;
```

```
    cout << "s2 = "<< s2;

    return 0;
}
```

Example: Reverse a sentence using recursion.
****

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

// function prototype
void reverse(const string& a);

int main() {
  string str;

  cout << " Please enter a string " << endl;
  getline(cin, str);

  // function call
  reverse(str);

  return 0;
}

// function definition
void reverse(const string& str) {

  // store the size of the string
  size_t numOfChars = str.size();

  if(numOfChars == 1) {
    cout << str << endl;
  }
  else {
    cout << str[numOfChars - 1];

    // function recursion
    reverse(str.substr(0, numOfChars - 1));
  }
}
```

Example 1: Concatenate String Objects
****

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

int main()
{
   string s1, s2, result;

   cout << "Enter string s1: ";
   getline (cin, s1);

   cout << "Enter string s2: ";
   getline (cin, s2);

   result = s1 + s2;

   cout << "Resultant String = "<< result;

   return 0;
}
```

Example 2: Concatenate C-style Strings
****

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

int main()
{
   char s1[50], s2[50];

   cout << "Enter string s1: ";
   cin.getline(s1, 50);

   cout << "Enter string s2: ";
   cin.getline(s2, 50);

   strcat(s1, s2);

   cout << "s1 = " << s1 << endl;
   cout << "s2 = " << s2;
```

```
    return 0;
}
```

Program 1: to illustrate non-static members
****

```
// C++ program to illustrate
// non-static data members
using namespace std;
#include <iostream>

// Class
class GfG {
private:
        // Created a variable
        int count = 0;

public:
        // Member function to increment
        // value of count
        void set_count()
        {
                count++;
        }

        // Member function to access the
        // private members of this class
        void show_count()
        {

                // print the count variable
                cout << count << '\n';
        }
};

// Driver Code
int main()
{
        // Objects of class GfG
        GfG S1, S2, S3;

        // Set count variable to 1
        // for each object
```

```
	S1.set_count();
	S2.set_count();
	S3.set_count();

	// Function to display count
	// for each object
	S1.show_count();
	S2.show_count();
	S3.show_count();

	return 0;
}
```

Program 2: to illustrate static members:
****

```
// C++ program to illustrate
// non-static data members
using namespace std;
#include <iostream>

// Class
class GfG {
private:
	// Created a static variable
	static int count;

public:
	// Member function to increment
	// value of count
	void set_count()
	{
		count++;
	}

	// Member function to access the
	// private members of this class
	void show_count()
	{

		// print the count variable
		cout << count << '\n';
```

```
        }
};

int GfG::count = 0;

// Driver Code
int main()
{
        // Objects of class GfG
        GfG S1, S2, S3;

        // Increment count variable
        // by 1 for each object
        S1.set_count();
        S2.set_count();
        S3.set_count();

        // Function to display count
        // for each object
        S1.show_count();
        S2.show_count();
        S3.show_count();

        return 0;
}
```

Dynamic Memory Allocation for Objects

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

#include <iostream>
using namespace std;

class Box {
  public:
    Box() {
      cout << "Constructor called!" <<endl;
    }
    ~Box() {
      cout << "Destructor called!" <<endl;
    }
};
int main() {
  Box* myBoxArray = new Box[4];
```

```
   delete [] myBoxArray; // Delete array

   return 0;
}
```

malloc example: random string generator

```
#include <stdio.h>      /* printf, scanf, NULL */
#include <stdlib.h>     /* malloc, free, rand */

int main ()
{
  int i,n;
  char * buffer;

  printf ("How long do you want the string? ");
  scanf ("%d", &i);

  buffer = (char*) malloc (i+1);
  if (buffer==NULL) exit (1);

  for (n=0; n<i; n++)
    buffer[n]=rand()%26+'a';
  buffer[i]='\0';

  printf ("Random string: %s\n",buffer);
  free (buffer);

  return 0;
}
```

****

1) C++ Recursion Example

```
#include<iostream>
using namespace std;
int main()
{
int factorial(int);
```

```
int fact,value;
cout<<"Enter any number: ";
cin>>value;
fact=factorial(value);
cout<<"Factorial of a number is: "<<fact<<endl;
return 0;
}
int factorial(int n)
{
if(n<0)
return(-1); /*Wrong value*/
if(n==0)
return(1);  /*Terminating condition*/
else
{
return(n*factorial(n-1));
}
}
```

Direct Recursion #

If a function calls itself, it's known as direct recursion. This results in a one-step recursive call: the function makes a recursive call inside its own function body.

****

Below is an example of a direct recursive function that computes the square of a number:

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

// recursive function to calculate square of a number
int square(int x)
 {
  // base case
    if (x == 0)
    {
      return x;
    }

  // recursive case
  else
   {
      return square(x-1) + (2*x) - 1;
   }
}
```

```
int main() {
  // implementation of square function
  int input=30;
  cout << input<<"^2 = "<<square(input);
  return 0;
}
```

****

Indirect Recursion
If the function f1 calls another function f2 and f2 calls f1 then it is indirect recursion (or mutual recursion).

This is a two-step recursive call: the function calls another function to make a recursive call.

```
#include <iostream>
using namespace std;
int n=0;
// declaring functions
void foo1(void);
void foo2(void);

// defining recursive functions
void foo1()
{
  if (n <= 20)
  {
    cout<<n<<" ";  // prints n
    n++;           // increments n by 1
    foo2();       // calls foo2()
  }
  else
    return;
}

void foo2()
{
  if (n <= 20)
  {
    cout<<n<<" ";  // prints n
    n++;           // increments n by 1
    foo1();       // calls foo1()
  }
  else
```

```
    return;
}

// Driver Program
int main(void)
{
  foo1();
  return 0;
}
```

****

Example 1: Fibonacci Series up to n number of terms

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

int main()
{
    int n, t1 = 0, t2 = 1, nextTerm = 0;

    cout << "Enter the number of terms: ";
    cin >> n;

    cout << "Fibonacci Series: ";

    for (int i = 1; i <= n; ++i)
    {
        // Prints the first two terms.
        if(i == 1)
        {
            cout << " " << t1;
            continue;
        }
        if(i == 2)
        {
            cout << t2 << " ";
            continue;
        }
        nextTerm = t1 + t2;
        t1 = t2;
        t2 = nextTerm;

        cout << nextTerm << " ";
    }
```

```
    return 0;
}
```

Example 2: Program to Generate Fibonacci Sequence Up to a Certain Number
****

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

int main()
{
    int t1 = 0, t2 = 1, nextTerm = 0, n;

    cout << "Enter a positive number: ";
    cin >> n;

    // displays the first two terms which is always 0 and 1
    cout << "Fibonacci Series: " << t1 << ", " << t2 << ", ";

    nextTerm = t1 + t2;

    while(nextTerm <= n)
    {
        cout << nextTerm << ", ";
        t1 = t2;
        t2 = nextTerm;
        nextTerm = t1 + t2;
    }
    return 0;
}


// CPP program To calculate The Value Of nCr


#include <bits/stdc++.h>
using namespace std;

int fact(int n);

int nCr(int n, int r)
{
        return fact(n) / (fact(r) * fact(n - r));
}
```

```
// Returns factorial of n
int fact(int n)
{
	int res = 1;
	for (int i = 2; i <= n; i++)
		res = res * i;
	return res;
}

// Driver code
int main()
{
	int n = 5, r = 3;
	cout << nCr(n, r);
	return 0;
}
```

Program for Tower of Hanoi
****

```
// C++ recursive function to
// solve tower of hanoi puzzle

#include <bits/stdc++.h>
using namespace std;

void towerOfHanoi(int n, char from_rod,
                                    char to_rod, char aux_rod)
{
	if (n == 1)
	{
		cout << "Move disk 1 from rod " << from_rod <<
                                                    " to rod " << to_rod<<endl;
		return;
	}
	towerOfHanoi(n - 1, from_rod, aux_rod, to_rod);
	cout << "Move disk " << n << " from rod " << from_rod <<
                                                        " to rod " << to_rod << endl;
	towerOfHanoi(n - 1, aux_rod, to_rod, from_rod);
}
```

```
// Driver code
int main()
{
        int n = 4; // Number of disks
        towerOfHanoi(n, 'A', 'C', 'B'); // A, B and C are names of rods
        return 0;
}
```

Example 1: Find GCD using while loop

****

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

int main()
{
   int n1, n2;

   cout << "Enter two numbers: ";
   cin >> n1 >> n2;

   while(n1 != n2)
   {
      if(n1 > n2)
         n1 -= n2;
      else
         n2 -= n1;
   }

   cout << "HCF = " << n1;
   return 0;
}
```

Example: 2. Find HCF/GCD using for loop

****

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

int main() {
   int n1, n2, hcf;
   cout << "Enter two numbers: ";
```

```
    cin >> n1 >> n2;

    // Swapping variables n1 and n2 if n2 is greater than n1.
    if ( n2 > n1) {
        int temp = n2;
        n2 = n1;
        n1 = temp;
    }

    for (int i = 1; i <=  n2; ++i) {
        if (n1 % i == 0 && n2 % i ==0) {
            hcf = i;
        }
    }

    cout << "HCF = " << hcf;
    return 0;
}

// C++ program to illustrate Ackermann function
#include <iostream>
using namespace std;

int ack(int m, int n)
{
        if (m == 0){
                return n + 1;
        }
        else if((m > 0) && (n == 0)){
                return ack(m - 1, 1);
        }
        else if((m > 0) && (n > 0)){
                return ack(m - 1, ack(m, n - 1));
        }
}

// Driver code
int main()
{
        int A;
        A = ack(1, 2);
        cout << A << endl;
        return 0;
}
```

Example: Display Largest Element of an array
****

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

int main()
{
   int i, n;
   float arr[100];

   cout << "Enter total number of elements(1 to 100): ";
   cin >> n;
   cout << endl;

   // Store number entered by the user
   for(i = 0; i < n; ++i)
   {
      cout << "Enter Number " << i + 1 << " : ";
      cin >> arr[i];
   }

   // Loop to store largest number to arr[0]
   for(i = 1;i < n; ++i)
   {
      // Change < to > if you want to find the smallest element
      if(arr[0] < arr[i])
          arr[0] = arr[i];
   }
   cout << "Largest element = " << arr[0];

   return 0;
}
```

Simple Searching In Array Example
****

```
#include <iostream>
#include<conio.h>

using namespace std;
```

```
#define ARRAY_SIZE 5

int main()
{
   int numbers[ARRAY_SIZE], i ,search_key;

         cout<<"Simple C++ Example Program for Simple Searching In Array\n";

         // Read Input
   for (i = 0; i < ARRAY_SIZE; i++)
   {
                  cout<<"Enter the Number : "<< (i+1) <<"  : ";
      cin>>numbers[i];
   }

   cout<<"Enter the key\n";
   cin>>search_key;

   /*  Simple Search with Position */
         for (i = 0; i < ARRAY_SIZE; i++)
   {
                  if(numbers[i] == search_key)
                  {
                           cout<<"Search Element Found . Position Is :"<< (i+1) <<" \n";
                           break;
                  }
                  if(i == ARRAY_SIZE - 1)
                  {
                           cout<<"Search Element is not in Array.\n";
                  }
   }


}
```