Table of Contents

All codes are uploaded into ideone.com

1	https://ideone.com/TXurpJ	6.ii	https://ideone.com/xt6c5M
2	https://ideone.com/Qab1qj	7.i	https://ideone.com/gXf2NT
3.i	https://ideone.com/kykJSJ	7.ii	https://ideone.com/7XcF4z
3.ii	https://ideone.com/HrQjwR	7.iii	https://ideone.com/vL8oMT
4.i	https://ideone.com/TnWtYR	8	https://ideone.com/Dq3III
4.ii	https://ideone.com/fqvGZd	9	https://ideone.com/SNT2M3
5	https://ideone.com/GSAkeg	10	https://ideone.com/Ix9lMB
6.i	https://ideone.com/f371ov	11	https://ideone.com/d3oF5C

Program: 1. Calculate the sum of the series $1^2+3^2+5^2+......+(2n+1)^2$

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
    int n, i, s;
    cin >> n;
    s = 0;
    for (i = 1; i <= n; i = i + 2) {
        s = s + i * i;
    }
    cout << s << "\n";
    return 0;
}
```

Input:

```
10
```

Output:

```
165
```

Program: 2. Write a Program to calculate the CGPA of a semester

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
    int t, n;
    float b, c = 0, e = 0, s, g;
    string a, d;
    cout << "== CGPA Calculator ==" << "\n";
    cout << "Total Course(s) = " ;
    {
        cin >> t;
        while (t--) {
            cout << "Course Title = ";
            cin >> d ;
            cout << "Credit(s) = ";
            cin >> c;
            cout << "Marks = ";
            cin >> n;
            {
                if (n < 40)
                    a = "F", b = 0.00;
                if (n >= 40 && n <= 44)
                    a = "D", b = 2.00;
                if (n >= 45 && n <= 49)
                    a = "C", b = 2.25;
                if (n >= 50 && n <= 54)
                    a = "C+", b = 2.50;
                if (n >= 55 && n <= 59)
                    a = "B-", b = 2.75;
                if (n >= 60 && n <= 64)
                    a = "B", b = 3.00;
                if (n >= 65 && n <= 69)
                    a = "B+", b = 3.25;
                if (n >= 70 && n <= 74)
                    a = "A-", b = 3.50;
                if (n >= 75 && n <= 79)
                    a = "A", b = 3.75;
                if (n >= 80 && n <=100)
                    a = "A+", b = 4.00;
            }
            e += c * b;
            cout << d << " Course: " << "Latter
Grade = " << a << " , " << "Grade Point = " <<
b << "\n";
            cout << "\n";
        }
    }
    cout << "Total Credits = ";
    cin >> s ;
    g = e / s;
    cout << "CGPA = " << fixed << setprecision
(2) << g;
    return 0;
}
```

Input:

```
3
CA
3
78
CA_Lab
2
72
Math_IV
3
66
8
```

Output:

```
== CGPA Calculator ==
Total Course(s) = 3
Course Title = CA
Credit(s) = 3
Marks = 78
CA Course: Latter Grade = A , Grade
Point = 3.75

Course Title = CA_Lab
Credit(s) = 2
Marks = 72
CA_Lab Course: Latter Grade = A- ,
Grade Point = 3.5

Course Title = Math_IV
Credit(s) = 3
Marks = 66
Math_IV Course: Latter Grade = B+ ,
Grade Point = 3.25

Total Credits = 8
CGPA = 3.50
```

Program: 3(i). Implement the *recursive* algorithm to find the factorial n

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

int Fact (int n)
{
    if (n == 0)
        return 1;
    else
        return (n * Fact (n - 1));
}

int main()
{
    int a, n;
    cout << "n = ";
    cin >> n;
    a = Fact (n);
    cout <<"factorial("<< n <<") = "<< a <<"\n";
    return 0;
}
```

Input:

```
5
```

Output:

```
n = 5
factorial(5) = 120
```

Program: 3(ii). Implement the *iterative* algorithm to find the factorial n

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

int main()
{
    int a, n, i;
    cout << "n = ";
    cin >> n;
    {
        if (n == 0)
            cout << "factorial(" << n << ") = "
<< 1 << "\n";
        else {
            a = 1;
            for (i = 1; i <= n; i++)
                a *= i;
    cout <<"factorial("<< n <<") = "<< a <<"\n";
        }
    }
    return 0;
}
```

Input:

```
5
```

Output:

```
n = 5
factorial(5) = 120
```

Program: 4(i). Implement the *recursive* algorithm to find the *n*th Fibonacci series

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

int Fib (int x)
{
    int f;
    if (x == 1)
        return (0);
    else if (x == 2)
        return (1);
    else
        f = Fib (x - 1) + Fib (x - 2);
    return (f);
}
int main()
{
    int Fib (int) ;
    int x, y, n;
    cout << "n = ";
    cin >> n;
    for (x = 1; x <= n; x++) {
        y = Fib (x);
        cout << y << "\t";
    }
    return 0;
}
```

Input:

```
6
```

Output:

```
n = 6
0     1     1     2     3     5
```

Program: 4(ii). Implement the *iterative* algorithm to find the *n*th Fibonacci series

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

int main()
{
    int n, i, a = 0, b = 1, s = 0;
    cout << "n = ";
    cin >> n;
    cout << a << "\t" << b << "\t";
    for (i = 2; i < n; ++i) {
        s = a + b;
        a = b;
        b = s;
        cout << s << "\t";
    }
    return 0;
}
```

Input:

```
6
```

Output:

```
n = 6
0     1     1     2     3     5
```

Program: 5. Implement the Towers of Hanoi algorithm

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

void TH (int, char, char, char);

int main()
{
    int n;
    cout << "Number of disk = ";
    cin >> n;
    TH (n, 'A', 'C', 'B');
}

void TH (int n, char x, char y, char z)
{
    if (n > 0) {
        TH (n - 1, x, z, y);
        cout << x << " => " << y << "\n";
        TH (n - 1, z, y, x);
    }
}
```

Input:

```
3
```

Output:

```
Number of disk = 3
A => C
A => B
C => B
A => C
B => A
B => C
A => C
```

Program: 6(i). Implement the Pizza Cutting algorithm by using *recursive* algorithm

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

int pizza (int n)
{
    if (n == 1)
        return 2;
    else if (n > 1)
        return pizza (n - 1) + n;
}

int main()
{
    int n;
    cout << "Cut = ";
    cin >> n;
    cout << "Piece = " << pizza (n) << "\n";
    return 0;
}
```

Input:

```
3
```

Output:

```
Cut = 3
Piece = 7
```

Program: 6(ii). Implement the Pizza Cutting algorithm by using *iterative* algorithm

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

int main()
{
    int n, s;
    cout << "Cut = ";
    cin >> n;
    if (n == 1)
        cout << "2" << "\n";
    else if (n > 1) {
        s = (1 + n * (n + 1) / 2);
        cout << "Piece = " << s << "\n";
    }
    return 0;
}
```

Input:

```
3
```

Output:

```
Cut = 3
Piece = 7
```

Program: 7(i). Calculate the series m^2+$(m+1)^2$+......+$(n-1)^2$ + n^2 by using *going-up* recursive algorithm

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

int sqsum (int m, int n)
{
    if (m < n) {
        return (m * m + sqsum (m + 1, n));
    }
    else
        return (m * m);
}

int main()
{
    int a, m, n;
    cout << "Values of m & n = ";
    cin >> m >> n;
    a = sqsum (m, n);
    cout << "Sum = " << a << "\n";
    return 0;
}
```

Input:

```
4 8
```

Output:

```
Values of m & n = 4 8
Sum = 190
```

Program: 7(ii). Calculate the series $m^2+(m+1)^2+......+(n-1)^2 + n^2$ by using *going-down* recursive algorithm

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

int sqsum (int m, int n)
{
    if (m < n) {
        return (sqsum (m, n - 1) + n * n);
    }
    else
        return (n * n);
}

int main()
{
    int a, m, n;
    cout << "Values of m & n = ";
    cin >> m >> n;
    a = sqsum (m, n);
    cout << "Sum = " << a << "\n";
    return 0;
}
```

Input:

```
4 8
```

Output:

```
Values of m & n = 4 8
Sum = 190
```

Program: 7(iii). Calculate the series $m^2+(m+1)^2+......+(n-1)^2 + n^2$ by using *splitting-halves* recursive algorithm

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

int sqsum (int m, int n)
{
    int mid = (m + n) / 2;
    if (m == n) {
        return (m * m);
    }
    else
        return(sqsum (m, mid)+sqsum (mid+1,n));
}

int main()
{
    int a, m, n;
    cout << "Values of m & n = ";
    cin >> m >> n;
    a = sqsum (m, n);
    cout << "Sum = " << a << "\n";
    return 0;
}
```

Input:

```
4 8
```

Output:

```
Values of m & n = 4 8
Sum = 190
```

Program: 8. Implement the Insertion Sort algorithm

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

int main()
{
    int size, arr_sort[100], i, j, a, t;
    cout << "Size of array = ";
    cin >> size;
    cout << "\n" << size << " Array elements for
sorting = " << "\n";
    for (i = 0; i < size; i++)
        cin >> arr_sort[i];
    cout << "\nElements = ";
    for (i = 0; i < size; i++) {
        cout << "\t" << arr_sort[i];
    }
    for (i = 1; i < size; i++) {
        t = arr_sort[i];
        j = i - 1;
        while (j >= 0 && arr_sort[j] > t) {
            arr_sort[j + 1] = arr_sort[j];
            j = j - 1;
        }
        arr_sort[j + 1] = t;
        cout << "\nSwap : " << i << " = ";
        for (a = 0; a < size; a++) {
            cout << "\t" << arr_sort[a];
        }
    }
    cout << "\n\nSorted   = ";
    for (i = 0; i < size; i++) {
        cout << "\t" << arr_sort[i];
    }
    return 0;
}
```

Input:

```
8
35 61 28 55 34 69 71 45
```

Output:

```
Size of array = 8

8 Array elements for sorting =
35 61 28 55 34 69 71 45

Elements =  35   61   28   55   34   69   71   45
Swap : 1 =  35   61   28   55   34   69   71   45
Swap : 2 =  28   35   61   55   34   69   71   45
Swap : 3 =  28   35   55   61   34   69   71   45
Swap : 4 =  28   34   35   55   61   69   71   45
Swap : 5 =  28   34   35   55   61   69   71   45
Swap : 6 =  28   34   35   55   61   69   71   45
Swap : 7 =  28   34   35   45   55   61   69   71

Sorted   =  28   34   35   45   55   61   69   71
```

Program: 9. Implement the Selection Sort algorithm

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

int main()
{
    int size, arr_sort[100], i, j, a, t, p;
    cout << "Size of array = ";
    cin >> size;
    cout << "\n" << size << " Array elements for
sorting = " << "\n";
    for (i = 0; i < size; i++)
        cin >> arr_sort[i];
    cout << "\nElements = ";
    for (i = 0; i < size; i++) {
        cout << "\t" << arr_sort[i];
    }
    for (i = 0; i < size; i++) {
        p = i;
        for (j = i; j < size; j++) {
            if (arr_sort[p] > arr_sort[j])
                p = j;
        }
        if (p != 1) {
            t = arr_sort[i];
            arr_sort[i] = arr_sort[p];
            arr_sort[p] = t;
        }
        arr_sort[j + 1] = t;
        cout << "\nSwap : " << i << " = ";
        for (a = 0; a < size; a++) {
            cout << "\t" << arr_sort[a];
        }
    }
    cout << "\n\nSorted   = ";
    for (i = 0; i < size; i++) {
        cout << "\t" << arr_sort[i];
    }
    return 0;
}
```

Input:

```
8
35 61 28 55 34 69 71 45
```

Output:

```
Size of array = 8

8 Array elements for sorting =
35 61 28 55 34 69 71 45

Elements =   35   61   28   55   34   69   71   45
Swap : 0 =   28   61   35   55   34   69   71   45
Swap : 1 =   28   34   35   55   61   69   71   45
Swap : 2 =   28   34   35   55   61   69   71   45
Swap : 3 =   28   34   35   45   61   69   71   55
Swap : 4 =   28   34   35   45   55   69   71   61
Swap : 5 =   28   34   35   45   55   61   71   69
Swap : 6 =   28   34   35   45   55   61   69   71
Swap : 7 =   28   34   35   45   55   61   69   71

Sorted   =   28   34   35   45   55   61   69   71
```

Program: 10. Implement the Merge Sort algorithm

```
#include<bits/stdc++.h>
using namespace std;
int Merge (int A[], int p, int q, int r)
{
    int n1, n2, i, j, k;
    n1 = q - p + 1;
    n2 = r - q;
    int L[n1], R[n2];
    for (i = 0; i < n1; i++) {
        L[i] = A[p + i];
    }
    for (j = 0; j < n2; j++) {
        R[j] = A[q + j + 1];
    }
    i = 0, j = 0;
    for (k = p; i < n1 && j < n2; k++) {
        if (L[i] < R[j]) {
            A[k] = L[i++];
        }
        else {
            A[k] = R[j++];
        }
    }
    while (i < n1) {
        A[k++] = L[i++];
    }
    while (j < n2) {
        A[k++] = R[j++];
    }
}
int MergeSort (int A[], int p, int r)
{
    int q;
    if (p < r) {
        q = (p + r) / 2;
        MergeSort (A, p, q);
        MergeSort (A, q + 1, r);
        Merge (A, p, q, r);
    }
}
int main()
{
    int n, A[100], i;
    cout << "Size of array = ";
    cin >> n;
    cout << "\n" << n << " Array elements for
sorting = " << "\n";
    for (i = 0; i < n; i++)
        cin >> A[i];
    cout << "\nElements = ";
    for (i = 0; i < n; i++) {
        cout << "\t" << A[i];
    }
    MergeSort (A, 0, n - 1);
    cout << "\nSorted array =  ";
    for (i = 0; i < n; i++) {
        cout << A[i] << "\t";
    }
    return 0;
}
```

Input:

```
8
35 61 28 55 34 69 71 45
```

Output:

```
Size of array = 8

8 Array elements for sorting =
35 61 28 55 34 69 71 45

Elements =   35   61   28   55   34   69   71   45
Sorted   =   28   34   35   45   55   61   69   71
```

Program: 11. Implement the Quick Sort algorithm

```
#include<bits/stdc++.h>
using namespace std;
int Quick (int a[], int start, int end)
{
    int pivot = a[end];
    int pvi = start;
    int i, t;
    for (i = start; i < end; i++) {
        if (a[i] <= pivot) {
            t = a[i];
            a[i] = a[pvi];
            a[pvi] = t;
            pvi++;
        }
    }
    t = a[end];
    a[end] = a[pvi];
    a[pvi] = t;
    return pvi;
}
void Quicksort (int a[], int start, int end)
{
    if (start < end) {
        int pvi = Quick (a, start, end);
        Quicksort (a, start, pvi - 1);
        Quicksort (a, pvi + 1, end);
    }
}
int main()
{
    int n, a[100], i;
    cout << "Size of array = ";
    cin >> n;
    cout << "\n" << n << " Array elements for
sorting = " << "\n";
    for (i = 0; i < n; i++) {
        cin >> a[i];
    }
    cout << "\nElements = ";
    for (i = 0; i < n; i++) {
        cout << "\t" << a[i];
    }
    Quicksort (a, 0, n - 1);
    cout << "\nSorted array =  ";
    for (i = 0; i < n; i++) {
        cout << a[i] << "\t";
    }
    return 0;
}
```

Input:

```
8
35 61 28 55 34 69 71 45
```

Output:

```
Size of array = 8

8 Array elements for sorting =
35 61 28 55 34 69 71 45

Elements =  35  61  28  55  34  69  71  45
Sorted   =  28  34  35  45  55  61  69  71
```