Winter Winning Camp

Name: -Shivam Yadav

UID: - 22BCS15259

Section: - 620-A

Date: -19/12/24

VERY EASY

1) **Sum of Natural Numbers up to N**

```
#include <iostream>
using namespace std;
int main()
{
   int n;
   cin>>n;
   int sum =(n*(n+1))/2;
   cout<<sum;
}
```

OUTPUT: -

```
100
5050
```

2) **Print Multiplication Table of a Number**

```
#include <iostream>
using namespace std;
int main()
{
   int a;
   cin>>a;
   for(int i = 1;i <=10;i++)
   {
   cout<<a*i<<endl;

   }
```

```
    return 0;
}
```

```
7
7
14
21
28
35
42
49
56
63
70
```

EASY

3) **Count Digits in a Number**

```
#include <iostream>
using namespace std;
int main()
{
    int n,count = 0;
    int b=n;
    cin>>n;
    while (n != 0)
    {
        int a = n%10;
        n = n/10 ;
        count ++;
    }
    cout<<count;
    return 0;
}
```

OUTPUT: -

```
4568
4
```

4) **Reverse a Number**

```
#include <iostream>
using namespace std;
int main()
{
   int n, rev =0,count = 0;
   int b=n;
   cin>>n;
   while (n != 0)
   {
      int a = n%10;
      n = n/10 ;
      rev = rev*10 +a;
      count ++;
   }
   cout<<rev;
   return 0;
}
```

```
89645
54698
```

5) **Print Odd Numbers up to N**

```
#include<iostream>
using namespace std;
int main()
{
   int n;
   cout<<"INPUT NO. : ";
   cin>>n;
   for(int i = 1; i<=n; i =i+2)
   {
      cout<<i<<"\n";
   }
   return 0;
}
```

OUTPUT: -

```
INPUT NO. : 15
1
3
5
7
9
11
13
15
```

6) **Find the Sum of Digits of a Number**

```
#include <iostream>
using namespace std;
int main()
{
   int n, sum =0,count = 0;
   int b=n;
   cin>>n;
   while (n != 0)
   {
      int a = n%10;
      n = n/10 ;
      sum = sum +a;
      count ++;
   }
   cout<<sum;
   return 0;
}
```

OUTPUT: -

```
2654
17
```

MEDIUM

7) **Function Overloading for Calculating Area.**

```
#include <iostream>
using namespace std;
double area(double a)
{
```

```
    cout<<(3.14*a*a)<<endl;
    return 0;
}
int area(int a)
{
    cout<<(a*a)<<endl;
    return 0;
}
double area(double a,double b)
{
    cout<<((a*b)/2)<<endl;
    return 0;
}
int main()
{
    area(3);
    area(4);
    area(2.5,3.5);

    return 0;
}
```

OUTPUT: -

```
9
16
4.375
```

8) **Function Overloading with Hierarchical Structure.**

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

int calculateSalary(int stipend)
{
return stipend;
}
int calculateSalary(int baseSalary, int bonuses)
{
```

```
return baseSalary + bonuses;
}
int calculateSalary(int baseSalary, int bonuses, int incentives)
{
return baseSalary + bonuses + incentives;
}
int main()
 {
int stipend, baseSalary, bonuses, incentives;
cin >> stipend;
cout << "Intern Salary: " << calculateSalary(stipend) << endl;
cin >> baseSalary >> bonuses;
cout << "Employee Salary: " << calculateSalary(baseSalary, bonuses) <<
endl;
cout << "Enter base salary, bonuses, and incentives for a manager: ";
cin >> baseSalary >> bonuses >> incentives;
cout << "Manager Salary: " << calculateSalary(baseSalary, bonuses,
incentives) << endl;
return 0;
}
```

OUTPUT: -

```
20000
Intern Salary: 20000
70000
60000
Employee Salary: 130000
Enter base salary, bonuses, and incentives for a manager: 100000
20000
10000
Manager Salary: 130000
```

9) **Matrix Multiplication Using Function Overloading**

```
#include <iostream>
#include <vector>
using namespace std;
vector<vector<int>> operate(const vector<vector<int>>& a, const
vector<vector<int>>& b) {
vector<vector<int>> result(a.size(), vector<int>(a[0].size()));
for (int i = 0; i < a.size(); i++) {
for (int j = 0; j < a[0].size(); j++) {
```

```
result[i][j] = a[i][j] + b[i][j];
}
}
return result;
}
vector<vector<int>> operate(const vector<vector<int>>& a, const
vector<vector<int>>& b, int) {
vector<vector<int>> result(a.size(), vector<int>(b[0].size(), 0));
for (int i = 0; i < a.size(); i++) {
for (int j = 0; j < b[0].size(); j++) {
for (int k = 0; k < a[0].size(); k++) {
result[i][j] += a[i][k] * b[k][j];
}
}
}
return result;
}
void displayMatrix(const vector<vector<int>>& matrix) {
for (const auto& row : matrix) {
for (int elem : row) {
cout << elem << " ";
}
cout << endl;
}
}
int main() {
int rows1, cols1, rows2, cols2;
cout << "Enter rows and columns for first matrix: ";
cin >> rows1 >> cols1;
cout << "Enter rows and columns for second matrix: ";
cin >> rows2 >> cols2;
if ((rows1 != rows2 || cols1 != cols2) && cols1 != rows2) {
cout << "Matrix operation not possible due to incompatible
dimensions!"
<< endl;
return 0;
}
```

```
vector<vector<int>> mat1(rows1, vector<int>(cols1));
vector<vector<int>> mat2(rows2, vector<int>(cols2));
cout << "Enter elements for first matrix:" << endl;
for (int i = 0; i < rows1; i++) {
for (int j = 0; j < cols1; j++) {
cin >> mat1[i][j];
}
}
cout << "Enter elements for second matrix:" << endl;
for (int i = 0; i < rows2; i++) {
for (int j = 0; j < cols2; j++) {
cin >> mat2[i][j];
}
}

if (rows1 == rows2 && cols1 == cols2) {
cout << "Matrix Addition Result:" << endl;
displayMatrix(operate(mat1, mat2)); // Add matrices
} else {
cout << "Matrix addition not possible (dimensions do not match)." <<
endl;
}

if (cols1 == rows2) {
cout << "Matrix Multiplication Result:" << endl;
displayMatrix(operate(mat1, mat2, 0)); // Multiply matrices
} else {
cout << "Matrix multiplication not possible (columns of first matrix
mustequal rows of second matrix)." << endl;
}
return 0;
}
```

OUTPUT:-

```
Enter rows and columns for first matrix: 2
2
Enter rows and columns for second matrix: 2
2
Enter elements for first matrix:
4
9
8
7
Enter elements for second matrix:
6
7
5
3
Matrix Addition Result:
10 16
13 10
Matrix Multiplication Result:
69 55
83 77
```

10) Implementing Polymorphism for Shape Hierarchies.

```
#include <iostream>
#include <cmath>
using namespace std;
class Shape {
public:
virtual void area() = 0;
virtual ~Shape() {}
};
class Circle : public Shape {
double radius;
public:
Circle(double r) : radius(r) {}
void area() override {
cout << "Area of Circle: " << M_PI * radius * radius << endl;
}
};
```

```
class Rectangle : public Shape {
double length, width;
public:
Rectangle(double l, double w) : length(l), width(w) {}
void area() override {
cout << "Area of Rectangle: " << length * width << endl;
}
};
class Triangle : public Shape {
double base, height;
public:
Triangle(double b, double h) : base(b), height(h) {}
void area() override {
cout << "Area of Triangle: " << 0.5 * base * height << endl;
}
};
int main() {
Shape* shapes[] = { new Circle(7), new Rectangle(7, 7), new Triangle(7,
7)
};
for (auto shape : shapes) {
shape->area();
delete shape;
}
return 0;
}
```

OUTPUT: -

```
Area of Circle: 153.938
Area of Rectangle: 49
Area of Triangle: 24.5
```