Name: Yash Kumar

UID: 22BCS15424

Section: 22BCS_IOT-620

1) Sum of Natural Numbers up to N

Code:

```
#include <iostream>
using namespace std;
int main() {
  int n;
  cout<<"enter"<<endl;
  cin>>n;
  int ans=0;
  for(int i=0;i<=n;i++){
      ans+=i;
  }
cout<<ans;
    return 0;
}
```

Output:

```cpp
// Online C++ compiler to run C++ program online
#include <iostream>
using namespace std;
int main() {
  int n;
  cout<<"enter"<<endl;
  cin>>n;
  int ans=0;
  for(int i=0;i<=n;i++){
      ans+=i;
  }
cout<<ans;
    return 0;
}
```

2) Check if a Number is Prime

Code:

```cpp
#include <iostream>
using namespace std;
int main() {
  int n;
  int flag=0;
  cout<<"enter"<<endl;
  cin>>n;
  for(int i=2;i<=n/2;i++){
    if(n%i==0){
      flag=1;
    }
  }
  if(flag==0){
    cout<<" prime"<<endl;
  }
```

```
  else{
     cout<<"not prime"<<endl;
  }

   return 0;
}
```

Output:

```
// Online C++ compiler to run C++ program online
#include <iostream>
using namespace std;
int main() {
    int n;
    int flag=0;
    cout<<"enter"<<endl;
    cin>>n;
    for(int i=2;i<=n/2;i++){
        if(n%i==0){
            flag=1;
        }
    }
    if(flag==0){
        cout<<" prime"<<endl;
    }
    else{
        cout<<"not prime"<<endl;
    }

     return 0;
}
```

3) Check if a Number is a Palindrome

Code:

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

```
int main() {
  int n;
  cout<<"enter"<<endl;
  cin>>n;
  int temp=n;
  int reverse=0;
  while(temp!=0){
     reverse=(reverse*10)+(temp%10);
     temp=temp/10;
  }
  if(reverse==n){
     cout<<"yes";
  }
  else{
     cout<<"no";
  }
   return 0;
}
```

Output:

main.cpp

```cpp
// Online C++ compiler to run C++ program online
#include <iostream>
using namespace std;

int main() {
    int n;
    cout<<"enter"<<endl;
    cin>>n;
    int temp=n;
    int reverse=0;
    while(temp!=0){
        reverse=(reverse*10)+(temp%10);
        temp=temp/10;
    }
    if(reverse==n){
        cout<<"yes";
    }
    else{
        cout<<"no";
    }
     return 0;
}
```

4)Function Overloading for Calculating Area.

Code:

```cpp
#include <iostream>
using namespace std;
void sum(int a,int b){
   int c=a*b;
   cout<<c<<endl;
}
void sum(double a,double b){
```

```
    double c=a*b;
    cout<<c<<endl;

}
void sum(int a,double b){
    double c=a*b;
    cout<<c<<endl;

}


int main() {
    sum(5,4);
    sum(5.5,6.5);
    sum(4,8.8);
    return 0;
}
```

Output:

main.cpp

```cpp
// Online C++ compiler to run C++ program online
#include <iostream>
using namespace std;
void sum(int a,int b){
    int c=a*b;
    cout<<c<<endl;
}
void sum(double a,double b){
    double c=a*b;
    cout<<c<<endl;

}
void sum(int a,double b){
    double c=a*b;
    cout<<c<<endl;

}


int main() {
    sum(5,4);
    sum(5.5,6.5);
    sum(4,8.8);
    return 0;
}
```

5) Function Overloading with Hierarchical Structure.

Objective

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

class Employee {
public:
```

```
    virtual void calculateSalary() {
        cout << "Calculating salary for an employee." << endl;
    }
};

class Intern : public Employee {
private:
    double stipend;
public:
    Intern(double stipend) : stipend(stipend) {}

    void calculateSalary() override {
        cout << "Intern's salary (stipend): $" << stipend << endl;
    }
};

class RegularEmployee : public Employee {
protected:
    double baseSalary;
    double bonuses;
public:
    RegularEmployee(double baseSalary, double bonuses)
        : baseSalary(baseSalary), bonuses(bonuses) {}

    void calculateSalary() override {
        cout << "Regular Employee's salary: $" << baseSalary + bonuses << endl;
    }
};

class Manager : public RegularEmployee {
private:
    double performanceIncentives;
public:
    Manager(double baseSalary, double bonuses, double performanceIncentives)
        : RegularEmployee(baseSalary, bonuses), performanceIncentives(performanceIncentives) {}

    void calculateSalary() override {
        cout << "Manager's salary: $" << (baseSalary + bonuses + performanceIncentives) << endl;
    }
};

int main() {
    Intern intern(1000);
    RegularEmployee regularEmployee(3000, 500);
    Manager manager(5000, 1000, 2000);

    intern.calculateSalary();
    regularEmployee.calculateSalary();
    manager.calculateSalary();

    return 0;
```

}
Output:

main.cpp

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

class Employee {
public:
    virtual void calculateSalary() {
        cout << "Calculating salary for an employee." << endl;
    }
};

class Intern : public Employee {
private:
    double stipend;
public:
    Intern(double stipend) : stipend(stipend) {}

    void calculateSalary() override {
        cout << "Intern's salary (stipend): $" << stipend << en
    }
};

class RegularEmployee : public Employee {
protected:
    double baseSalary;
    double bonuses;
public:
```

6) Polymorphism with Shape Area Calculation.

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

```
class Shape {
public:
   virtual double calculateArea() = 0;
   virtual ~Shape() {}
};

class Circle : public Shape {
private:
   double radius;
public:
   Circle(double r) : radius(r) {}

   double calculateArea() override {
      return M_PI * radius * radius;
   }
};

class Rectangle : public Shape {
private:
   double length, breadth;
public:
   Rectangle(double l, double b) : length(l), breadth(b) {}

   double calculateArea() override {
      return length * breadth;
   }
};

class Triangle : public Shape {
private:
   double base, height;
public:
   Triangle(double b, double h) : base(b), height(h) {}

   double calculateArea() override {
      return 0.5 * base * height;
   }
};

int main() {
   double radius, length, breadth, base, height;

   cout << "Enter the radius of the circle: ";
   cin >> radius;

   cout << "Enter the length and breadth of the rectangle: ";
   cin >> length >> breadth;

   cout << "Enter the base and height of the triangle: ";
   cin >> base >> height;
```

```
    Shape* circle = new Circle(radius);
    Shape* rectangle = new Rectangle(length, breadth);
    Shape* triangle = new Triangle(base, height);

    cout << "Area of Circle: " << circle->calculateArea() << endl;
    cout << "Area of Rectangle: " << rectangle->calculateArea() << endl;
    cout << "Area of Triangle: " << triangle->calculateArea() << endl;

    delete circle;
    delete rectangle;
    delete triangle;

    return 0;
}
```

Output:

main.cpp

```
    cout << "Enter the radius of the circle: ";
    cin >> radius;

    cout << "Enter the length and breadth of the rectangle: "
    cin >> length >> breadth;

    cout << "Enter the base and height of the triangle: ";
    cin >> base >> height;

    Shape* circle = new Circle(radius);
    Shape* rectangle = new Rectangle(length, breadth);
    Shape* triangle = new Triangle(base, height);

    cout << "Area of Circle: " << circle->calculateArea() <<
    cout << "Area of Rectangle: " << rectangle->calculateArea
        endl;
    cout << "Area of Triangle: " << triangle->calculateArea()
        endl;

    delete circle;
    delete rectangle;
```

7) Inheritance with Student and Result Classes.

Code:

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

class Student {
protected:
   int rollNumber;
   string name;
public:
   void getStudentDetails() {
```

```
        cout << "Enter Roll Number: ";
        cin >> rollNumber;
        cin.ignore();
        cout << "Enter Name: ";
        getline(cin, name);
    }

    void displayStudentDetails() {
        cout << "Roll Number: " << rollNumber << endl;
        cout << "Name: " << name << endl;
    }
};

class Result : public Student {
private:
    float marks1, marks2, marks3;
    float totalMarks;
    float percentage;
public:
    void getMarks() {
        cout << "Enter marks for Subject 1: ";
        cin >> marks1;
        cout << "Enter marks for Subject 2: ";
        cin >> marks2;
        cout << "Enter marks for Subject 3: ";
        cin >> marks3;
    }

    void calculateResult() {
        totalMarks = marks1 + marks2 + marks3;
        percentage = (totalMarks / 300) * 100;
    }

    void displayResult() {
        displayStudentDetails();
        cout << "Total Marks: " << totalMarks << "/300" << endl;
        cout << "Percentage: " << percentage << "%" << endl;
    }
};

int main() {
    Result studentResult;

    studentResult.getStudentDetails();
    studentResult.getMarks();
    studentResult.calculateResult();
    studentResult.displayResult();

    return 0;
}
```

Output:

main.cpp

```
        cin >> marks3;
    }

    void calculateResult() {
        totalMarks = marks1 + marks2 + marks3;
        percentage = (totalMarks / 300) * 100;
    }

    void displayResult() {
        displayStudentDetails();
        cout << "Total Marks: " << totalMarks << "/300" << endl
        cout << "Percentage: " << percentage << "%" << endl;
    }
};

int main() {
    Result studentResult;

    studentResult.getStudentDetails();
    studentResult.getMarks();
    studentResult.calculateResult();
    studentResult.displayResult();

    return 0;
```

8) Function Overloading for Complex Number Operations.

Code:

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

class Complex {
```

```
private:
    double real;
    double imag;

public:
    Complex() : real(0), imag(0) {}
    Complex(double r, double i) : real(r), imag(i) {}

    void input() {
        cout << "Enter real part: ";
        cin >> real;
        cout << "Enter imaginary part: ";
        cin >> imag;
    }

    void display() const {
        if (imag >= 0)
            cout << real << " + " << imag << "i" << endl;
        else
            cout << real << " - " << -imag << "i" << endl;
    }

    Complex operator+(const Complex& c) {
        return Complex(real + c.real, imag + c.imag);
    }

    Complex operator*(const Complex& c) {
        return Complex(real * c.real - imag * c.imag, real * c.imag + imag * c.real);
    }

    double magnitude() const {
        return sqrt(real * real + imag * imag);
    }
};

int main() {
    int choice;
    Complex c1, c2, result;

    cout << "Complex Number Operations\n";
    cout << "1. Addition\n";
    cout << "2. Multiplication\n";
    cout << "3. Magnitude\n";
    cout << "Enter your choice (1/2/3): ";
    cin >> choice;

    if (choice == 1 || choice == 2) {
        cout << "Enter first complex number:\n";
        c1.input();
        cout << "Enter second complex number:\n";
        c2.input();
```

```
    } else if (choice == 3) {
        cout << "Enter a complex number:\n";
        c1.input();
    }

    switch (choice) {
        case 1:
            result = c1 + c2;
            cout << "Sum: ";
            result.display();
            break;
        case 2:
            result = c1 * c2;
            cout << "Product: ";
            result.display();
            break;
        case 3:
            cout << "Magnitude of the complex number: " << fixed << setprecision(2) << c1.magnitude()
<< endl;
            break;
        default:
            cout << "Invalid choice\n";
    }

    return 0;
}
```

Output:

main.cpp

Share

```
        cout << "Enter a complex number:\n";
        c1.input();
    }

    switch (choice) {
        case 1:
            result = c1 + c2;
            cout << "Sum: ";
            result.display();
            break;
        case 2:
            result = c1 * c2;
            cout << "Product: ";
            result.display();
            break;
        case 3:
            cout << "Magnitude of the complex number: " << fi
                setprecision(2) << c1.magnitude() << endl;
            break;
        default:
            cout << "Invalid choice\n";
    }

    return 0;
}
```

9)Print Multiplication Table of a Number

Code:

```
// Online C++ compiler to run C++ program online
#include <iostream>
using namespace std;
int main() {
  int n;
```

```
 cout<<"enter"<<endl;
 cin>>n;
 int ans=0;
 for(int i=1;i<=10;i++){
    ans=n*i;
    cout<<n<<"*"<<i<<"= " <<ans<<endl;
 }

  return 0;
}
```

Output:

```
#include <iostream>
using namespace std;
int main() {
  int n;
  cout<<"enter"<<endl;
  cin>>n;
  int ans=0;
  for(int i=1;i<=10;i++){
      ans=n*i;
      cout<<n<<"*"<<i<<"= " <<ans<<endl;
  }

    return 0;
}
```

10)) Count Digits in a Number

Code:

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

int main() {
 int n;
 cout<<"enter"<<endl;
 cin>>n;
 int count=0;
 while(n>0){
    n=n/10;
    count++;
 }
cout<<count;
  return 0;
```

}

Output:

```
// Online C++ compiler to run C++ program online
#include <iostream>
using namespace std;

int main() {
  int n;
  cout<<"enter"<<endl;
  cin>>n;
  int count=0;
  while(n>0){
      n=n/10;
      count++;
  }
cout<<count;
    return 0;
}
```