Question 1-

CODE-

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

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

   if (n > 0) {
      int totalSum = n * (n + 1) / 2;
      cout << "The sum of natural numbers from 1 to " << n << " is: " <<
totalSum << endl;
   } else {
      cout << "Please enter a positive integer." << endl;
   }
return 0;
}
```

Output-

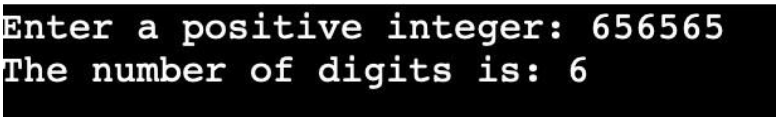
```
Enter a positive integer: 24
The sum of natural numbers from 1 to 24 is: 300
```

Question 2-

CODE-

```
#include <iostream> using
namespace std;
int    main()
{ int  n;  int
count = 0;
   cout << "Enter a positive integer: ";
cin >> n;    if (n <= 0) {
       cout << "Please enter a positive integer." << endl;
return 0;
   }     while (n !=
0) {         n = n /
10;
count++;
   }
   cout << "The number of digits is: " << count << endl;
return 0;
}
```

Output-



Question 3-

CODE-

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

```
double calculateArea(double radius)
{ return M_PI * radius * radius;
}
double calculateArea(double length, double breadth)
{ return length * breadth;
}
double calculateArea(double base, double height, bool isTriangle)
{ return 0.5 * base * height;
}

int main()
{ double radius;
   cout << "Enter the radius of the circle: ";
cin >> radius;
   cout << "Area of the circle: " << calculateArea(radius) << endl;

   double length, breadth;
   cout << "Enter the length and breadth of the rectangle: ";
cin >> length >> breadth;
   cout << "Area of the rectangle: " << calculateArea(length, breadth) << endl;

   double base, height;
   cout << "Enter the base and height of the triangle: ";
cin >> base >> height;
   cout << "Area of the triangle: " << calculateArea(base, height, true) << endl;

   return 0;
}
```

Output-

```
Enter the radius of the circle: 20
Area of the circle: 1256.64
Enter the length and breadth of the rectangle: 2 4
Area of the rectangle: 8
Enter the base and height of the triangle: 4 8
Area of the triangle: 16
```

Ouestion 4-

CODE-

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

class Account { protected:
   double balance;

public:
   Account(double bal) : balance(bal) {}
virtual void calculateInterest() = 0;       virtual
~Account() {}
};

class SavingsAccount : public Account
   { double rate;
int time;
public:
   SavingsAccount(double bal, double r, int t) : Account(bal), rate(r), time(t) {}

   void calculateInterest() override {
      double interest = balance * (rate / 100) * time;          cout
<< "Savings Account Interest: " << interest << endl;
      cout << "Total Balance after Interest: " << (balance + interest) << endl;
   }
};

class CurrentAccount : public Account
{ double maintenanceFee;

public:
   CurrentAccount(double bal, double fee) : Account(bal), maintenanceFee(fee)
{}

   void calculateInterest() override {
```

```
        double finalBalance = balance - maintenanceFee;          cout << "Current
Account Maintenance Fee: " << maintenanceFee << endl;          cout << "Final
Balance after Fee Deduction: " << finalBalance << endl;
    }
};

int main() {     int
accountType;
    cout << "Enter Account Type (1 for Savings, 2 for Current): ";
cin >> accountType;

    if (accountType == 1)
{ double balance, rate;
        int time;
        cout << "Enter Balance: ";
cin >> balance;
        cout << "Enter Interest Rate (%): ";
cin >> rate;
        cout << "Enter Time (years): ";
        cin >> time;

        SavingsAccount savings(balance, rate, time);
        savings.calculateInterest();       }
else if (accountType == 2)
{  double balance, maintenanceFee;
cout << "Enter Balance: ";          cin
>> balance;
        cout << "Enter Monthly Maintenance Fee: ";
cin >> maintenanceFee;

        CurrentAccount current(balance, maintenanceFee);
        current.calculateInterest();
    } else {
        cout << "Invalid account type entered." << endl;
    }

    return 0;
}
```

Output-

```
Enter Account Type (1 for Savings, 2 for Current): 1
Enter Balance: 20000
Enter Interest Rate (%): 22
Enter Time (years):
4
Savings Account Interest: 17600
Total Balance after Interest: 37600
```

Question 5-

CODE-

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

class Employee
{ protected:
string name;
int id;     double
salary;

public:
   Employee(string empName, int empId, double empSalary)
      : name(empName), id(empId), salary(empSalary) {}

   virtual void calculateEarnings() = 0;
virtual void displayDetails() {         cout
<< "Name: " << name << endl;          cout
<< "ID: " << id << endl;
      cout << "Base Salary: " << salary << endl;
   }

   virtual ~Employee() {}
};
```

```
class Manager : public Employee
{ int performanceRating;

public:
   Manager(string empName, int empId, double empSalary, int rating)
      : Employee(empName, empId, empSalary), performanceRating(rating) {}

   void calculateEarnings() override
{ double bonus = 0;      switch
(performanceRating) {            case 5:
bonus = salary * 0.2; break;            case 4:
bonus = salary * 0.15; break;             case 3:
bonus = salary * 0.1; break;            default:
bonus = 0; break;
      }
      cout << "Performance Rating: " << performanceRating << endl;
cout << "Bonus: " << bonus << endl;
      cout << "Total Earnings: " << (salary + bonus) << endl;
   }
};

class Developer : public Employee
   { int extraHours;

public:
    Developer(string empName, int empId, double empSalary, int hours)
       : Employee(empName, empId, empSalary), extraHours(hours) {}

   void calculateEarnings() override {         double overtimeCompensation =
extraHours * 50;         cout << "Extra Hours Worked: " << extraHours <<
endl;        cout << "Overtime Compensation: " << overtimeCompensation <<
endl;        cout << "Total Earnings: " << (salary + overtimeCompensation) <<
endl;
   }
};

int main() {     int
employeeType;
   cout << "Enter Employee Type (1 for Manager, 2 for Developer): ";
cin >> employeeType;
```

```
   string name;
int id;     double
salary;

   cout << "Enter Name: ";
cin >> name;     cout <<
"Enter ID: ";     cin >> id;
   cout << "Enter Salary: ";
cin >> salary;

   if (employeeType == 1)
{ int performanceRating;
       cout << "Enter Performance Rating (1-5): ";
cin >> performanceRating;

       Manager manager(name, id, salary, performanceRating);
manager.displayDetails();          manager.calculateEarnings();      } else if
(employeeType == 2) {
       int extraHours;
       cout << "Enter Extra Hours Worked: ";
       cin >> extraHours;

       Developer developer(name, id, salary, extraHours);
       developer.displayDetails();
developer.calculateEarnings();
   } else {
       cout << "Invalid Employee Type!" << endl;
   }

   return 0;
}
```

Output-

```
Enter Employee Type (1 for Manager, 2 for Developer): 2
Enter Name: Vansh
Enter ID: 12
Enter Salary: 50000
Enter Extra Hours Worked: 8
Name: Vansh
ID: 12
Base Salary: 50000
Extra Hours Worked: 8
Overtime Compensation: 400
Total Earnings: 50400
```