Unit 1 easy questions

1.

Write a program to convert a string into upper or lower case in C++ using classes? And also capital count
Enter the string: Good morning all
The string in upper case: GOOD MORNING ALL The string in lower case: good morning all
1. She is Good
2. OK!
3. 1234tellme
4.0000.000
5.Take.on.me

Sol:

```
#include <iostream>

#include <cctype>

#include <string>

using namespace std;

int main()

{

  char str[100];

  cout << "Enter a string: ";

  gets(str);

  int count=0;

  for(int i = 1; i < 100; i++)

  {

    if(str[i] >= 'A' && str[i] <= 'Z')

    {

          count++;

          }

  }
```

```
 cout << "total uppercase letters in a string are: "<<count-1<<endl;
 for(char &c : str)
 {
  c= toupper(c);
 }
 cout<<"uppercase string: "<<str<<endl;
 for(char &c : str)
 {
  c=tolower(c);
 }
 cout<<"lowercase string: "<<str<<endl;
 return 0;
}
```

2.Write a C++ program to find even or odd numbers using any conditional statement?

```
#include <iostream>
int main()
 {
   int number;
   cout << "Enter a number: ";
   cin >> number;
   if (number % 2 == 0) {
       cout << number << " is an even number." << std::endl;
   } else {
       cout << number << " is an odd number." << std::endl;
   }
   return 0;
}
```

3. Find if a number is prime or not using conditional statements in C++.

```
#include <iostream>
int main()
 {
   int num;
   std::cout << "Enter a number: ";
   std::cin >> num;
   if (num <= 1) {
      std::cout << num << " is not a prime number." << std::endl;
      return 0;
   }
   bool isPrime = true;
   for (int i = 2; i < num; ++i) {
      if (num % i == 0) {
         isPrime = false; // If the number is divisible by any other number, it's not prime
         break;
      }
   }
   if (isPrime) {
      std::cout << num << " is a prime number." << std::endl;
   } else {
      std::cout << num << " is not a prime number." << std::endl;
   }
   return 0;
}
```

4. Write a Program to convert Celsius to Fahrenheit in C++?

```
#include <iostream>
```

```
using namespace std;

int main()

 {

   float celsius, fahrenheit;

   cout << "Enter the temperature in Celsius: ";

   cin >> celsius;

   fahrenheit = (celsius * 9.0 / 5.0) + 32.0;

   cout << "The temperature in Fahrenheit is: " << fahrenheit << "°F" << endl;

   return 0;

}
```

5. Write a program in C++ to check if a number is palindrome or not using control statements?

```
#include <iostream>

int main()

 {

   int number, originalNumber, reversedNumber = 0, remainder;

   std::cout << "Enter a number: ";

   std::cin >> number;

   originalNumber = number; // Store the original number for comparison

   while (number > 0) {

     remainder = number % 10;

       reversedNumber = reversedNumber * 10 + remainder;

       number /= 10;

   }


   if (originalNumber == reversedNumber) {

       std::cout << "The number is a palindrome." << std::endl;
```

```
    } else {
        std::cout << "The number is not a palindrome." << std::endl;
    }
    return 0;
}
```

Unit 2 easy questions

1.

Develop a c++ program for default arguments
Enter the vlaue: 10 15 25 30
80

```
#include <iostream>
int sum(int a = 0, int b = 0, int c = 0, int d = 0) {
    return a + b + c + d;
}

int main() {
    int num1, num2, num3, num4;
   std::cout << "Enter four numbers (separated by spaces): ";
    std::cin >> num1 >> num2 >> num3 >> num4;
    int result = sum(num1, num2, num3, num4);
    std::cout << "Sum of the numbers is: " << result << std::endl;
    return 0;
}
```

2.

Develop a program to check the entered user name is valid or not using function . Get both the inputs from the user.
Enter the user name: Saveetha@789

Reenter the user name: Saveetha@123
User name is Invalid

```
#include <iostream>
#include <string>
bool isValidUsername(const std::string& username1, const std::string& username2) {
    return username1 == username2;
}
int main() {
    std::string userName, reenteredName;
    std::cout << "Enter the user name: ";
    std::cin >> userName;
    std::cout << "Reenter the user name: ";
    std::cin >> reenteredName;
    if (isValidUsername(userName, reenteredName)) {
        std::cout << "User name is Valid" << std::endl;
    } else {
        std::cout << "User name is Invalid" << std::endl;
    }
    return 0;
}
```

3. **Develop** a program to find whether the person is eligible for vote or not. And if that particular person is not eligible, then print how many years are left to be eligible.

Note : Use the function **getperson** to get the input

```
#include <iostream>
void getAge(int &age) {
    std::cout << "Enter your age: ";
```

```
    std::cin >> age;
}
void checkEligibility(int age) {
    int votingAge = 18;
    if (age >= votingAge) {
        std::cout << "You are eligible to vote." << std::endl;
    } else {
        int yearsLeft = votingAge - age;
        std::cout << "You are not eligible to vote. Years left to be eligible: " << yearsLeft << "
year(s)." << std::endl;
    }
}
int main() {
    int personAge;
    getAge(personAge);
    checkEligibility(personAge);
    return 0;
}
```

4. **Develop** a program using function to calculate the simple interest. Suppose the customer is a senior citizen. He is being offered 12 percent rate of interest; for all other customers, the ROI is 10 percent.

```
#include <iostream>
// Function to calculate simple interest for senior citizen
float calculateInterestSenior(float principal, float rate, float time) {
    return (principal * rate * time) / 100;
}
// Function to calculate simple interest for other customers
float calculateInterest(float principal, float time) {
```

```
    float rate = 10.0; // Default rate for other customers
    return (principal * rate * time) / 100;
}
int main() {
    float principal, time, interest;
    std::cout << "Enter principal amount: ";
    std::cin >> principal;
    std::cout << "Enter time period (in years): ";
    std::cin >> time;
    char customerType;
    std::cout << "Are you a senior citizen? (Y/N): ";
    std::cin >> customerType;
    if (customerType == 'Y' || customerType == 'y') {
        interest = calculateInterestSenior(principal, 12.0, time);
    } else {
        interest = calculateInterest(principal, time);
    }
    std::cout << "Simple interest: " << interest << std::endl;
    return 0;
}
```

5. **Develop** a program using choice to check given string in palindrome or not using inline function.

```
#include <iostream>
#include <algorithm>
using namespace std;
inline bool isPalindrome(const string& str) {
    return str == string(str.rbegin(), str.rend());
```

```
}
int main() {
   string inputString;
      cout << "Enter a string: ";
   cin >> inputString;
if (isPalindrome(inputString)) {
      cout << "The string is a palindrome." << endl;
   } else {
      cout << "The string is not a palindrome." << endl;
   }
   return 0;
}
```

Unit 3 easy questions.

1.Write C++ Program to display the cube of the number up to a given integer using Destructor

```
#include<iostream>
using namespace std;
class cube
{
        public:
                int i,n;
                cube()
                {
                        cout<<"enter the value ";
                        cin>>n;
                        for(i=1;i<=n;i++)
                        {
```

```
                cout<<i*i*i<<endl;
            }
        }
    ~cube ()
    {
        cout<<"object is destroyed";
    }
};
int main()
{
    cube c;
}
```

2. Write C++ Program to display the cube of the number up to a given integer using uctor overloading

```
#include<iostream>
using namespace std;
class cube
{
    public:
        int i,n;
        cube()
        {
            cout<<"enter the value ";
            cin>>n;
            for(i=1;i<=n;i++)
            {
                cout<<i*i*i<<endl;
```

```
                }
        }
};
int main()
{
    cube c;
}
```

3. Write a program in C++ to find the sum of the series using the constructor overloading.

```
#include<iostream>
using namespace std;
class sum
{
    public:
        int i,n,count=0;
        sum()
        {
            cout<<"enter the value ";
            cin>>n;
            for(i=1;i<=n;i++)
            {
                count=count+i;
            }
            cout<<count;
        }
};
int main()
{
```

```
    sum s;
}
```

4. Write a program in C++ to print a pattern of right angle triangle with a number that will repeat a number in the row by using the constructor overloading.

```
#include<iostream>
using namespace std;
class pattern
{
    public:
        int num,i,j;
        pattern()
        {
            cout<<"enter the number of rows";
            cin>>num;
            for(i=1;i<=num;i++)
            {
                for(j=1;j<=i;j++)
                {
                    cout<<j<<" ";
                }
                cout<<" \n";
            }
        }
        ~pattern()
        {
            cout<<"object is destroyed";
        }
```

```
};
int main()
{
	pattern p;
	return 0;
}
```

6.	Write a C++ Program to display the reverse of a number using the constructor overloading

```
#include<iostream>
using namespace std;
class reverse
{
	public:
		int n,rem,rev=0;
		reverse()
		{
			cout<<"enter the value ";
			cin>>n;
			while(n!=0)
			{
				rem=n%10;
				rev=rev*10+rem;
				n=n/10;
			}
			cout<<rev;
		}
```

```
};
int main()
{
	reverse s;
}
```

6. Write a C++ program of binary to octal conversion with Constructor with constructor.

```
#include<iostream>
using namespace std;
class convert
{
	public:
		int bin,i,j=1,oct=0;
		convert()
		{
			cout<<"enter the binary number ";
			cin>>bin;
			while(bin>0)
			{
				i=bin%2;
				oct=oct+i*j;
				j=j*2;
				bin=bin/8;
			}
			cout<<oct;
		}
};
int main()
```

```
{
        convert s;
}
```

8. Write a C++ program to find the number and sum of all integer between 100 and 200 which are divisible by 9 with constructor destructor.

```
#include<iostream>
using namespace std;
class divisible
{
        public:
                int i,count=0;
                divisible()
                {
                        for(i=100;i<=200;i++)
                        {
                                if(i%9==0)
                                {
                                        count++;
                                        cout<<i<<endl;
                                }
                   }
                   cout<<count<<endl;
                }
};
int main()
{
        divisible d;
```

}

Unit 4

1.2C++ code to print the given values in program using pointer to object
1,54
The real part is 1 The imaginary part is 54

```
#include <iostream>
struct ComplexNumber
{
   int real;
   int imaginary;
};
int main()
{
   ComplexNumber number;
   std::cout << "Enter the real part: ";
   std::cin >> number.real;
   std::cout << "Enter the imaginary part: ";
   std::cin >> number.imaginary;
   ComplexNumber *ptr = &number;
   std::cout << "The real part is " << ptr->real << std::endl;
   std::cout << "The imaginary part is " << ptr->imaginary << std::endl;
   return 0;
}
```

2.Build a C++ code to find the area of a rectangle using the concept of array of objects.

```
#include <iostream>
```

```
struct Rectangle {
    double length;
    double width;
    double getArea() const { return length * width; }
};
int main() {
    Rectangle rectangle;
    std::cout << "Enter length of the rectangle: ";
    std::cin >> rectangle.length;
    std::cout << "Enter width of the rectangle: ";
    std::cin >> rectangle.width;
    double area = rectangle.getArea();
    std::cout << "Area of the rectangle: " << area << std::endl;
    return 0;
}
```

3.Write a C++ program to find the sum of two numbers using the concept of C++ multiple inheritance.

```
#include<iostream>
using namespace std;
class number1
{
        public:
                int n1;
                void getdata1()
                {
                        cout<<"Enter the first number :";
                        cin>>n1;
```

```
                }
};
class number2
{
        public:
                int n2;
                void getdata2()
                {
                        cout<<"Enter the second number :";
                        cin>>n2;
                }
};
class sum:public number1, public number2
{
        public:
                int sum;
                void getdata3()
                {
                        sum=n1+n2;
                        cout<<"The sum of the two numbers is :"<<sum;
                }
};
int main()
{
        sum s;
        s.getdata1();
        s.getdata2();
```

```
        s.getdata3();
}
```

4.

C++ Program to to display address of each element of an array
No input is needed
Displaying address using arrays: &arr[0] = 0x61fef0 &arr[1] = 0x61fef4 &arr[2] = 0x61fef8 Displaying address using pointers: ptr + 0 = 0x61fef0 ptr + 1 = 0x61fef4 ptr + 2 = 0x61fef8

```
#include <iostream>
using namespace std;
int main()
{
   int arr[] = {10, 20, 30};
   int *ptr = arr;
   cout << "Displaying address using arrays: \n";
   for (int i = 0; i < 3; ++i)
        {
      cout << "&arr[" << i << "] = " << &arr[i] <<endl;
   }
   cout << "\nDisplaying address using pointers: \n";
```

```
    for (int i = 0; i < 3; ++i)
            {
        cout << "ptr + " << i << " = " << ptr + i <<endl;
    }
  return 0;
}
```

5.C++ Program that illustrates how to use ‘this’ pointer.
The input given in the program itself
x = 20

```
#include <iostream>
class Sample
{
private:
    int x;
public:
    void setX(int x)
            {
        this->x = x;
    }
 int getX() const {
        return x;
    }
};
int main()
{
    Sample obj;
```

```
    obj.setX(20);
    std::cout << "x = " << obj.getX() << std::endl;
    return 0;
}
```

6.

Write a C++ code to find area of square and circle using abstract class and pure virtual function
Enter radius of the circle: 5 Enter the length of the square: 4
Area of square: 16 Area of circle: 78.5

```
 #include<iostream>
#include<cmath>
using namespace std;
class calculator
{
        public:
                int area_of_square(int side)
                {
                        return side*side;
                }
                int area_of_circle(double radius)
                {
                        return 3.14*radius*radius;
                }
```

```
};
int main()
{
        calculator calc;
        int side=3;
        double radius=5;
        cout<<calc.area_of_square(side)<<endl;
        cout<<calc.area_of_circle(radius)<<endl;
        return 0;
}
```

7.

Build a C++ code to print the address of the variable.
Input given in the program itself
Address of var1: 0x7fff5fbff8ac Address of var2: 0x7fff5fbff8a8 Address of var3: 0x7fff5fbff8a4

```
#include <iostream>
Using namespace std;
int main() {
   int var1 = 10;
   int var2 = 20;
   int var3 = 30;
   cout << "Address of var1: " << &var1 << endl;
   cout << "Address of var2: " << &var2 << endl;
   cout << "Address of var3: " << &var3 << endl;
   return 0;
}
```

8.

Program to demonstrate multiple inheritance in c++
Input given in the program itself.
value of a: 100 value of b: 200 value of c: 300

```
#include <iostream>
using namespace std;
class A
{
public:
    int a;
    A() : a(100)
    {}
};
class B
{
public:
    int b;
    B() : b(200)
    {}
};
class C : public A, public B
{
public:
    int c;
```

```
    C() : c(300)
        {}
};
int main()
{
    C obj;
    cout << "value of a: " << obj.a << endl;
    cout << "value of b: " << obj.b << endl;
    cout << "value of c: " << obj.c << endl;
    return 0;
}
```

9.

C++ program to demonstrate example of private simple inheritance
value of x: 10
Here x is the protected data member of class A, class A is inherited privately to class B. In private inheritance only protected data member and member functions can be accessed by the derived class.

```
#include <iostream>
using namespace std;
class A
{
public:
        int x;
    A() : x(0) {}
    void setX(int value)
        {
```

```
        x = value;
    }
};
class B : public A
{
public:
    void display() {
        std::cout << "Value of x: " << x << std::endl;
    }
};
int main()
{
    B objB;
    objB.setX(10);
    objB.display();
    return 0;
}
```

10.

Unit 5 questions

Write a program to calculate the bonus of the employees. The class master derives the information from both admin and account classes which derives information from the class person. Create base and all derived classes having same member functions and parameters called getdata, display data and bonus.
enter salary : 10000
bonus = 11000

```
#include <iostream>
```

```
// Base class: Person
class Person {
protected:
    int salary;
public:
    void getData() {
        std::cout << "Enter salary: ";
        std::cin >> salary;
    }
    void displayData() const {
        std::cout << "Salary: " << salary << std::endl;
    }
    virtual void calculateBonus() = 0; // Virtual function for calculating bonus
};
// Derived class: Admin
class Admin : public Person {
public:
    void calculateBonus() override {
        int bonus = salary + 1000; // Example bonus calculation
        std::cout << "Bonus: " << bonus << std::endl;
    }
};
// Derived class: Account
class Account : public Person {
public:
    void calculateBonus() override {
        int bonus = salary + 1000; // Example bonus calculation
```

```
        std::cout << "Bonus: " << bonus << std::endl;
    }
};
// Derived class: Master
class Master : public Admin, public Account {
public:
    // Override calculateBonus to avoid ambiguity due to multiple inheritance
    void calculateBonus() override {
        int bonus = salary + 1000; // Example bonus calculation
        std::cout << "Bonus: " << bonus << std::endl;
    }
};
int main() {
    Master employee;
    // Get salary from user
    employee.getData();
    // Display salary and calculate bonus
    employee.displayData();
    employee.calculateBonus();
    return 0;
}
```

2. Write a C++ program to demonstrate multiple inheritance by creating a class cuboid which extends class rectangle, class shape. It calculates area and volume. Use appropriate member functions and member variables.

```
#include <iostream>
// Base class: Shape
class Shape {
protected:
```

```
    float length;
    float width;
public:
    void setDimensions(float l, float w) {
        length = l;
        width = w;
    }
};
// Base class: Rectangle
class Rectangle : public Shape {
public:
    float calculateArea() {
        return length * width;
    }
};
// Derived class: Cuboid (inherits from Rectangle and Shape)
class Cuboid : public Rectangle, public Shape {
private:
    float height;
public:
    void setDimensions(float l, float w, float h) {
        length = l;
        width = w;
        height = h;
    }
    float calculateVolume() {
        return length * width * height;
```

```
    }
};
int main() {
    Cuboid myCuboid;
    float length, width, height;
    std::cout << "Enter length, width, and height of the cuboid: ";
    std::cin >> length >> width >> height;
    myCuboid.setDimensions(length, width);
    myCuboid.setDimensions(length, width, height);
    std::cout << "Area of the base (Rectangle): " << myCuboid.calculateArea() << std::endl;
    std::cout << "Volume of the cuboid: " << myCuboid.calculateVolume() << std::endl;
    return 0;
}
```

3.

Create a class Number and derive the class Skipper from Number. Write a program to print the numbers from M to N by skipping K numbers in between? Use appropriate functions and variables.
M = 50 N = 100 K = 7
50, 58, 66, 74, …..

```
#include <iostream>
using namespace std;
class Number
{
```

```
public:
    int M, N, K;
    Number(int m, int n, int k) : M(m), N(n), K(k) {}
};
class Skipper : public Number
{
public:
    Skipper(int m, int n, int k) : Number(m, n, k) {}
    void printSkippingNumbers()
            {
        for (int i = M; i <= N; i += K)
            cout << i << " ";
        cout << std::endl;
    }
};
int main()
{
    int M, N, K;
    cout << "Enter M, N, and K: ";
    cin >> M >> N >> K;
    Skipper skipper(M, N, K);
    cout << "Numbers from " << M << " to " << N << " skipping " << K << " numbers in
between: ";
    skipper.printSkippingNumbers();
    return 0;
}
```

4.

A Grandfather is the owner of 500Cr property, he wants to give this property to his grandson. What are the possible ways to access this property by his grandson? Use appropriate functions and variables.
500
Received the Property

```
#include <iostream>
class Grandfather {
protected:
    int property;
public:
    Grandfather(int prop) : property(prop) {}
    int getProperty() { return property; }
};
class Grandson : public Grandfather {
public:
    using Grandfather::Grandfather; // Inheriting constructors
    void accessProperty() { std::cout << "Received the Property" << std::endl; }
};
int main() {
    int propertyValue;
    std::cout << "Enter the property value: ";
    std::cin >> propertyValue;
    Grandson grandson(propertyValue);
    (grandson.getProperty() == propertyValue) ? grandson.accessProperty() : std::cout << "Did not receive the full property" << std::endl;
    return 0;
```

}