Q1.>simple calculator

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

int main() {

  char op;
  float num1, num2;

  cout << "Enter operator: +, -, *, /: ";
  cin >> op;

  cout << "Enter two operands: ";
  cin >> num1 >> num2;

  switch(op) {

    case '+':
      cout << num1 << " + " << num2 << " = " << num1 + num2;
      break;

    case '-':
      cout << num1 << " - " << num2 << " = " << num1 - num2;
      break;

    case '*':
      cout << num1 << " * " << num2 << " = " << num1 * num2;
      break;

    case '/':
      cout << num1 << " / " << num2 << " = " << num1 / num2;
      break;

    default:
      // If the operator is other than +, -, * or /, error message is shown
      cout << "Error! operator is not correct";
      break;
  }

  return 0;
}
```

convert seconds in hh:mm:ss

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

int main()
{
   // declare variables
   int time = 0;
   int hour = 0;
   int min = 0;
   int sec = 0;

   // obtain data from user
   cout << "Enter a time in seconds: ";
   cin >> time;

   // using the time from ^ above, convert
   // secs to HH:MM:SS format using division
   // and modulus
   hour = time/3600;
   time = time%3600;
   min = time/60;
   time = time%60;
   sec = time;

   // display data to user
   cout<<"\nThe time in HH:MM:SS format is: "<<hour<<"
hours, "
       <<min<<" minutes, and "<<sec<<" seconds!\n";

   return 0;
}
```

```
#include <iostream>

using name space std;

void Sqvol (float side) {
 cout << " The volume of square is "<<< side << endl;
 }

void conevol (flat r, float h) {
cout << "The volume of the cone is " << 3-14*r*r*h/3 <<endl;
}

void rec vol (float I, float b , float h){
cout<<"The volume of the rectangle is " << l*b*h<<endl;
}

int main() {

float side, r, h1, l, w, h2;

cout << "Enter side to find volume of square!"<<endl;
cin>>side;

Sqvol (side);

cout<<"Entoy radius and height to find volume of come:" <<
endl;
cin>> r >> h1;
Cone vol (r, h);

cout << "Enter lenght, width and height to find volume of
rectangle?"<<endl;
cin >>l >>w>>h2;

recvol (l , w, h2);

return 0;
}
```

2.a)greatest of three number

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

int main() {

    double n1, n2, n3;

    cout << "Enter three numbers: ";
    cin >> n1 >> n2 >> n3;

    if(n1 >= n2 && n1 >= n3){
        cout << "Largest number: " << n1;
}
    else if(n2 >= n1 && n2 >= n3){
        cout << "Largest number: " << n2;
   }
    else {
        cout << "Largest number: " << n3;
  }
    return 0;
}
```

2.b)find the sum of even n odd

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

int main()
{
 int number, maximum, evenSum = 0, oddSum = 0;

 cout << "\nPlease Enter the Maximum Limit for Even & Odd
Numbers  =  ";
 cin >> maximum;

 for(number = 1; number <= maximum; number++)
 {
   if ( number % 2 == 0 )
  {
   evenSum = evenSum + number;
  }
  else
  {
   oddSum = oddSum + number;
  }
 }

 cout << "\nThe Sum of All Even Numbers upto " << maximum <<
" = " << evenSum;
 cout << "\nThe Sum of All Odd Numbers upto " << maximum <<
"  = " << oddSum;

  return 0;
}
```

2.c)prime nos b/w i and n

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

int main() {
    int n, upto;
    cout << "Enter the value of upto: ";
    cin >> upto;
    cout << "Prime numbers between 1 and " << upto<< ":" <<
endl;


   for(n=2; n<=upto; n++){
    for (int i = 2; i <=( n/2); i++) {
        if (n%i==0) {
          i=n;
          break;
        }
    }
    if(i!=n){
    cout << n<<"";
    }
  }
    return 0;
}
```

3.a)print name of the student n roll no.

```
#include <iostream>
#include <string>

using namespace std;

class Student {
   private:
      string name;
      int roll_number;

   public:
      void set_name(string student_name) {
         name = student_name;
      }

      void set_roll_number(int student_roll_number) {
         roll_number = student_roll_number;
      }

      string get_name() {
         return name;
      }

      int get_roll_number() {
         return roll_number;
      }
};

int main() {
   Student student;
   student.set_name("John");
   student.set_roll_number(12345);
   cout << "Name: " << student.get_name() << endl;
   cout << "Roll number: " << student.get_roll_number() << endl;
   return 0;
}
```

3.c)friend function

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

class a;
class b
{
   int number;
   public:
   b(int x)
   {
      number=x;
   }
   void friend greatest(a a1,b b1);
};

class a
{
   int number;
   public:
   a(int x)
   {
      number=x;
   }
   void friend greatest(a a1,b b1);
};
void greatest(a a1,b b1)
{
   if(a1.number>b1.number)
   {
      cout<<"\n Number in class A is greatest i.e. "<<a1.number;
   }
   else if(a1.number<b1.number)
   {
      cout<<"\n Number in class B is greatest i.e. "<<b1.number;
   }
   else
   {
      cout<<"\n Number in both classes are equal";
   }
}
int main()
{
   int num;

   cout<<"\n\n Enter number for class A - ";
   cin>>num;
   a a1(num);

   cout<<"\n Enter number for class B - ";
   cin>>num;
   b b1(num);

   greatest(a1,b1);
   cout<<"\n";

   return 0;
}
```

3.e) copy constructor

```
#include <iostream>

using namespace std;

class MyClass {
   private:
      int x;
   public:
      MyClass(int n) {
         x = n;
      }
      MyClass(const MyClass& other) {
         x = other.x;
         cout << "Copy constructor called" << endl;
      }

      int getX() {
         return x;
      }
};

int main() {
   MyClass obj1(5);
   MyClass obj2 = obj1; // Copy constructor called

   cout << "obj1.x = " << obj1.getX() << endl;
   cout << "obj2.x = " << obj2.getX() << endl;

   return 0;
}
```

4.a) complex Number

```
#include <iostream>

using namespace std;

class Complex {
  private:
    double real;
    double imag;

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

    Complex operator+(Complex const &obj) {
      Complex res;
      res.real = real + obj.real;
      res.imag = imag + obj.imag;
      return res;
    }

    Complex operator*(Complex const &obj) {
      Complex res;
      res.real = (real * obj.real) - (imag * obj.imag);
      res.imag = (real * obj.imag) + (imag * obj.real);
      return res;
    }

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

int main() {
  Complex c1(2, 3);
  Complex c2(4, 5);
  Complex c3 = c1 + c2;
  Complex c4 = c1 * c2;

  c1.print(); // Output: 2 + 3i
  c2.print(); // Output: 4 + 5i
  c3.print(); // Output: 6 + 8i
  c4.print(); // Output: -7 + 22i

  return 0;
}
```

3.b)new/delete operator

```
#include <iostream>

using namespace std;

class MyClass {
   private:
      int x;
      int y;

   public:
      MyClass() {
         cout << "Object created!" << endl;
      }

      void set_values(int a, int b) {
         x = a;
         y = b;
      }

      void print() {
         cout << "x: " << x << ", y: " << y << endl;
      }

      void *operator new(size_t size) {
         cout << "Overloading new operator with size: " << size << endl;
         void *p = ::new MyClass;
         return p;
      }

      void operator delete(void *p) {
         cout << "Overloading delete operator" << endl;
         free(p);
      }
};

int main() {
   MyClass *obj = new MyClass;
   obj->set_values(5, 10);
   obj->print();

   delete obj;

   return 0;
}
```

Single Inheritance

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

// Base class
class Vehicle {
  public:
    void drive() {
      cout << "Vehicle is driving." << endl;
    }
};

// Derived class
class Car : public Vehicle {
  public:
    void honk() {
      cout << "Car is honking." << endl;
    }
};

// Main function
int main() {
  // Creating object of derived class
  Car myCar;

  // Calling methods of base and derived class
  myCar.drive();
  myCar.honk();

  return 0;
}
```

Multiple inheritance

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

// Base class 1
class Vehicle {
  public:
    void drive() {
      cout << "Vehicle is driving." << endl;
    }
};

// Base class 2
class Radio {
  public:
    void playMusic() {
      cout << "Radio is playing music." << endl;
    }
};

// Derived class
class Car : public Vehicle, public Radio {
  public:
    void honk() {
      cout << "Car is honking." << endl;
    }
};

// Main function
int main() {
  // Creating object of derived class
  Car myCar;

  // Calling methods of base and derived class
  myCar.drive();
  myCar.playMusic();
  myCar.honk();

  return 0;
}
```

Multilevel

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

// Base class
class Vehicle {
  public:
    void drive() {
      cout << "Vehicle is driving." << endl;
    }
};

// Derived class 1
class Car : public Vehicle {
  public:
    void honk() {
      cout << "Car is honking." << endl;
    }
};

// Derived class 2
class SportsCar : public Car {
  public:
    void race() {
      cout << "Sports car is racing." << endl;
    }
};

// Main function
int main() {
  // Creating object of derived class
  SportsCar mySportsCar;

  // Calling methods of base and derived class
  mySportsCar.drive();
  mySportsCar.honk();
  mySportsCar.race();

  return 0;
}
```

Virtual function class

```
#include <iostream.h>
class Base{
   public:
   void display(){
      cout<<"Display base\n";
   }
   virtual void show(){
     cout<<"Show Base\n";
   }
};

class Derived : public Base{
   public:
   void display (){
    cout<<"Display Derived\n";
   }
   void show(){
     cout<<"Show Derived\n";
   }
};
int main() {
   Derived D;
   Base* ptr;
   cout<<"ptr points to base\n";

   ptr=&D;
   ptr->display();
   ptr->show();

   cout<<"ptr points to derived\n";
   ptr=&D;
   ptr->display();
   ptr->show();

   return 0;
}
```

self,unself preccision

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

int main()
{
   cout << "Example for formatted IO" << endl;

   cout << "Default: " << endl;
   cout << 123 << endl;

   cout << "width(5): " << endl;
   cout.width(5);
   cout << 123 << endl;

   cout << "width(5) and fill('*'): " << endl;
   cout.width(5);
   cout.fill('*');
   cout << 123 << endl;

   cout.precision(5);
   cout << "precision(5) ---> " << 123.4567890 << endl;
   cout << "precision(5) ---> " << 9.876543210 << endl;

   cout << "setf(showpos): " << endl;
   cout.setf(ios::showpos);
   cout << 123 << endl;

   cout << "unsetf(showpos): " << endl;
   cout.unsetf(ios::showpos);
   cout << 123 << endl;

   return 0;
}
```

6.a) name n age ptr

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

class person{
char name[20];
float age;
public:
person(char *s, float a){
strcpy(name, s);
age = a;
}
person &greater(person &x)
{
if(x.age> age){
return x;
}else{
return *this;
}
}
void display(void){
cout<<"Name:"<<name<<"\n"
     <<"Age: "<<age<<"\n";
}
};
int main(){
person pl("Pankaj",18.0),p2("Shahnawaz",23.0),p3("Roshan", 17.0);
person p=pl.greater (p2);
cout<<"Elder Person is:\n";
p.display();

p=pl.greater (p2);
cout<<"\nElder Person is;\n";
p.display();
return 0;
}
```

9.a) catch try

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

float division(int x,int y){

  if(y==0){
    throw"Attempted to divide by zero!";

 }
return (x/y);

}

int main(){
int i=25;
int j=0;

float k=0;
try{

  k=division(i,j);
  cout<<k<<endl;

 }

catch(const char*e){
   cerr<<e<<endl;

}
return 0;
}
```