

Prerequisite Programs

Q1 Write C++ program for addition of two numbers.

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

void main()
{
    int a, b, sum;
    cout << "Enter values of a&b" << endl;
    cin >> a >> b;
    sum = a+b;
    cout << "The addition is" << sum;
}
```

Q2 Write a C++ program to perform arithmetic operation using switch case

```
#include <iostream>
using namespace std;
void main()
{
    double num1, num2;
    char op;
    cout << "Enter two numbers" << endl;
    cin >> num1 >> num2;
    cout << "Enter operator : " << endl;
    cin >> op;
    switch (op)
    {
        case '+':
            cout << "Result" << (num1 + num2) << endl;
            break;
    }
}
```

```

    cout << "Result" << (num1 - num2) << endl;
    break;
}

case '#':
    cout << "Result" << (num1 * num2) << endl;
    break;

case '/':
    if (num1 != 0) cout << "Result" << (num1 / num2)
    else cout << "Error"
    break;

case '%':
    if (num1 != 0)
        cout << "Result" << (num1 % num2) << endl;
    break;

default:
    cout << "Invalid Operator!" << endl;
}
}

check_if_even_or_odd()
{
    #include <iostream.h>
    using namespace std;
    int main()
    {
        int num;
        cout << "Enter a number";
        cin >> num;
        if (num % 2 == 0) cout << "Even";
        else cout << "Odd";
    }
}

```

Q.4
1-10 using for loop
include <iostream>
using namespace std;

```

int main()
{
    for (int i = 1; i <= 10; i++)
        cout << i << " ";
}

```

Q.5
while loop 10-1
include <iostream>
using namespace std;

```

int main()
{
    int i = 10;
    while (i != 0)
    {
        cout << i << endl;
        i--;
    }
}

```

Q.6,7,8 pattern problems

— next page —

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

void pattern1(Cint n) {
    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= n - i + 1; j++) {
            cout << " ";
        }
        for (int j = 1; j <= i; j++) {
            cout << "* ";
        }
        cout << endl;
    }
}

void pattern2(Cint n) {
    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= i; j++) {
            cout << " " << "* ";
        }
        cout << endl;
    }
}

void pattern3(Cint n) {
    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= n - i + 1; j++) {
            cout << "j" << " ";
        }
        cout << endl;
    }
}
```

int main() {

```
    pattern1(5);
    pattern2(5);
    pattern3(5);
```

Q
19/9

Experiment 2

1)

```
class Student {  
    int roll_no;  
    String name;  
public:  
    void accept() {  
        cout << "Enter"  
        cin >> name >> roll_no;  
    }  
    void display() {  
        cout << "Name : " << name;  
        cout << "Roll : " << roll_no;  
    }  
};  
  
int main() {  
    Student S1;  
    S1.accept();  
    S1.display();  
    return 0;  
}
```

2)

```
#include <iostream>  
using namespace std;  
  
class book {  
public:  
    String b_name;  
    int b_price;  
    int b_pages;  
public:  
    void accept() {  
        cout << "Enter the data";  
        cin >> b_name >> b_price >> b_pages;  
    }  
    void display() {  
        cout << "Book Name : " << b_name << b_price << b_pages  
        << endl;  
    }  
};  
  
int main() {  
    book B1, B2;  
    B1.accept();  
    B2.accept();  
    if (B1.price > B2.price) B1.display();  
    else B2.display();  
}
```

3)

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

class timesec {
    int hours;
    char c;
public:
    void accept();
    cout << "Enter time in HH:MM:SS:"
        << endl;
    cin >> hours >> c >> minutes >> seconds;
};

void display();
int totalseconds = hours * 3600 + minutes * 60 + seconds;
cout << "Time in seconds: " << totalseconds << endl;
}

int main();
timesec t;
t.accept();
t.display();

3)
```

Ans

19/11

Experiment 3

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

class City {
public:
    int popul;
    string name;
};

void accept();
cout << "City: " << "Population: " << endl;
cin >> name >> popul;
};

void display();
cout << "Name: " << name << endl;
cout << "Population: " << popul << endl;
};

int main();
City c[5];
int max_pop = 0;
for (int i = 0; i < 5; i++) {
    c[i].accept();
}
for (int i = 0; i < 5; i++) {
    if (c[i].popul > (max_pop)) {
        max_pop = i;
    }
}
cout << "City with maximum population" << c[max_pop].display();
```

Q2

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

class account {
public:
    int acc_no;
    int balance;

    void accept() {
        cout << "Enter the details";
        cin >> acc_no >> balance;
    }

    void display() {
        cout << "Account No. " << acc_no << endl;
        if(balance > 1000) {
            cout << "Amount No. " << acc_no << endl;
            cout << "Balance : " << balance;
        }
        cout << "Acc no : " << acc_no << "balance : " << balance;
    }

    int main() {
        account a[10];
        for(int i=0; i<10; i++) {
            a[i].accept();
        }

        cout << "greater balance than 1000 : ";
        for(int i=0; i<10; i++) {
            a[i].display();
        }
    }
}
```

Q3

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

class Staff {
    string name;
    string post;
public:
    void accept() {
        cout << "Enter";
        getline(cin, name);
        getline(cin, post);
    }

    void display() {
        if(post == "HOD" || post == "Head") {
            cout << "Name : " << name;
            cout << "Post : " << post;
        }
    }
}
```

Q4

```
int main() {
    staff s[5];
    for(int i=0; i<5; i++) {
        s[i].accept();
    }

    for(int i=0; i<5; i++) {
        s[i].display();
    }
}
```

⑦
19/9

Experiment 2

```
Q1  
#include <iostream>  
using namespace std;  
  
class book {  
    string book_title;  
    string author_name;  
    float price;  
  
public:  
    void accept();  
    cout << "Enter book title, author & price";  
    cin >> book_title >> author_name >> price;  
};  
  
void display();  
cout << book_title << book_title;  
cout << "Author name" << author_name;  
cout << "Price" << price;  
}  
  
int main(){  
    Book b;  
    Book *ptr;  
    ptr = &b;  
    ptr->accept();  
    ptr->display();  
    return 0;  
}
```

```
Q2  
#include <iostream>  
using namespace std;  
  
class Student {  
    int roll_no;  
    int percentage;  
  
public:  
    void accept();  
    cout << "Enter data";  
    cin >> this->roll_no >> this->percentage;  
};  
  
void display();  
this->accept();  
cout << "Roll no." << roll_no;  
}  
  
int main(){  
    Student s;  
    Student *ptr;  
}
```

Q3 #include <iostream>
using namespace std;

class Student
public:
 class mark
 public:
 int m1, m2, m3
 void getinput()
 {
 cout << "m1: " << m1 << endl;
 cout << "m2: " << m2 << endl;
 cout << "m3: " << m3 << endl;
 }
 float getpercentage()
 {
 float total = (m1 + m2 + m3) / 3.0;
 return total;
 }
};

int main() {
 Student :: mark mark;
 mark.getinput();
 cout << "Percentage = " << mark.getpercentage();
 return 0;
}

8/10
19/10

Experiment 4

```

91 #include <iostream>
using namespace std;

class Number {
    int value1, value2;
public:
    void getvalues() {
        cout << "Enter value 1: ";
        cin >> value1;
        cout << "Enter value 2: ";
        cin >> value2;
    }

    void display() {
        cout << value1 << endl;
        cout << value2 << endl;
    }

    void swapValues(Number& obj) {
        int temp = obj.value1;
        obj.value1 = obj.value2;
        obj.value2 = temp;
    }
};

int main() {
    Number n1;
    n1.getvalues();
    cout << "Before Swapping: " << endl;
    n1.display();
    cout << "After Swapping: " << endl;
    n1.swapvalues();
    n1.display();
}

```

92 # Friend function

```

92 #include <iostream>
using namespace std;

class Number {
    int n1, n2;
public:
    void accept() {
        cout << "Enter the number: ";
        cin >> n1 >> n2;
    }

    void display() {
        cout << "n1 = " << n1 << endl;
        cout << "n2 = " << n2 << endl;
    }

    friend void swapNumber(Number &n);
};

void swapNumber(Number &n) {
    int temp = n.n1;
    n.n1 = n.n2;
    n.n2 = temp;
}

int main() {
    Number n;
    n.accept();
    cout << "Before swap: " << endl;
    n.display();
    swapNumber(n);
    cout << "After swap: " << endl;
    n.display();
}

```

Q3 WAP to swap 2 numbers from different classes using function

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

class num1{
    int n1;
public:
    void accept();
    cout<<"Enter number in class 1: ";
    cin>>n1;
}

void display();
cout<<"Class 1 "<<n1;

friend void swapNumber(num1&obj1, num2&obj2);

class num2{
    int n2;
public:
    void accept();
    cout<<"Enter the number in class 2: ";
    cin>>n2;
}

void display();
cout<<"Class 2 "<<n2;

friend void swapNumber(num1&obj1, num2&obj2);
```

```
void swapNumber (num1&obj1, num2&obj2)
    int temp = obj1.n1;
    obj1.n1 = obj2.n2;
    obj2.n2 = temp;
```

```
int main()
    num1 A;
    num2 B;
    A.accept();
    B.accept();
    cout << "Before swapping a & b are: " << a << b;
    A.display();
    B.display();
    swapNumber (A,B);
    cout << "After swap" << endl;
    A.display();
    B.display();
```

94

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

class Result1 {
    float mark1;
public:
    void accept() {
        cin >> mark1;
    }
    friend float Average(Result1, Result2);
};

class Result2 {
    float mark2;
public:
    void accept() {
        cout << "Enter marks for second student : ";
        cin >> mark2;
    }
    friend float Average (Result1, Result2);
};

float Average (Result1, Result2) {
    float avg = (mark1 + mark2) / 2;
    return avg;
}
```

int main() {

Result1 r1;

Result2 r2;

float avg;

r1.accept();

r2.accept();

avg = Average(r1, r2);
 cout << "Average of two results is : " << avg << endl;

}

Q) WAP to find greater number among two numbers from class using friend function.

1)

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

class Number {
public:
    int num;
    void accept() {
        cout << "Enter number: ";
        cin >> num;
    }
};

friend void findGreater(Number a, Number b);

2)
class Number2 {
public:
    int num;
    void accept() {
        cout << "Enter number: ";
        cin >> num;
    }
};

3)
void findGreater(Number a, Number b) {
    if(a.num == b.num) cout << "Equal " << a.num;
    else if(a.num > b.num) cout << "Greater no is " << a.num << endl;
    else cout << "Greater no is " << b.num << endl;
}
```

int main() {
 Number a;
 Number b;
 a.accept();
 b.accept();
 cout << findGreater(a, b);
 return 0;
}

Practical Questions

Q1
#include <iostream>
using namespace std;

class Number
{
 int value;

public:
 class C1A {
 int valueA;
 public:
 void input() {
 cout << "Enter value for Class A:";
 cin >> valueA;
 }
 };

 friend int sum(C1A a, C1B b);
};

Q2
class C1B {
 int valueB;
public:
 void input() {
 cout << "Enter value for Class B:";
 cin >> valueB;
 }

 friend int sum(C1A a, C1B b);
};
int sum(C1A a, C1B b) {
 return a.value + b.valueB;
}

int main(){
 C1A objA;
 C1B objB;

 objA.input();
 objB.input();

 cout << "Sum: " << sum(objA, objB) << endl;
 return 0;
}

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

class Number {
public:
    void accept();
    cout << "Enter the value: ";
    class value;
};

class value {
public:
    void display();
    cout << value << endl;
};

friend void swapNumber(Number &n1, Number &n2);
```

```
3: void swapNumber (Number&n1, Number&n2) {
    int temp = n1.value;
    n1.value = n2.value;
    n2.value = temp;
```

```
int main() {
    classA Number num1, num2;
    num1.accept();
    num2.accept();
    swapNumber(num1, num2);
    num1.display();
    num2.display();
```

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

class Student {
public:
    string name;
    int mark1, mark2, mark3;
};

public:
    Student (string n, int m1, int m2, int m3) {
        name = n;
        mark1 = m1;
        mark2 = m2;
        mark3 = m3;
    }

    friend void calculateAverage (Student);
```

```
3: void calculateAverage (Student) {
    float avg = (mark1 + mark2 + mark3) / 3;
    cout << "Student name: " << name << endl;
    cout << "Average marks: " << avg << endl;
```

```
int main() {
    Student s1 = ("Atikay", 90, 91, 92);
    calculateAverage (s1);
    return 0;
```

Q3
#include <iostream>
using namespace std;

class Box
double volume
public:

void input()
double length, width, height;
cout << "Enter length width height of a cube: ";
cin >> length >> width >> height;

volume = length * width * height;

3. friend void findGreater(Box b, cube c);

void findGreater(Box b, cube c);

{}
class Box2
double volume
public:

void input();

double length, width, height;
cout << "Enter length, width, height of a cube: ";
cin >> length >> width >> height;
volume = length * width * height;

3. friend void findGreater(Box b, cube c);

void findGreater(Box b, cube c);
b.volume > c.volume ? cout << "Box volume is greater"
: cout << "Cube volume is greater";
cout << "Box" << b.volume << endl << "Cube" << c.volume << endl;

3.
int main()

Box b;
cube c;
b.input();
c.input();
findGreater(b, c);

When we use more than one constructor in same class then it is called as constructor overloading.

i) write the C++ code to create the class rectangle having data member as L & B.

Calculate the area of rectangle for 3 objects. Use constructor overloading to initialize the object.

#include <iostream>

int main()

Ok
||||

Assignment #5

Cone-Shot (code)

#include <iostream>

#include <string>

using namespace std;

ii) a) Default:

class SumDefault

int n; long long sum;

public:

SumDefault()

cout << "Enter n : "

cin >> n;

sum = 0;

for(int i=1; i<=n; i++) sum = sum + i;

} //

void show() {cout << "Sum(" << n << ")" << endl;}

ii) b) Parameterized

class SumParam

int n; long long sum;

public:

SumParam(int N) :

n = N;

sum = 0;

for(int i=1; i<=n; i++) sum = sum + i;

} //

int getN() const {return n;}

```

// D. Copy
class StudentCopy {
    int m_lap;
    string name;
}

public:
    StudentCopy(string name);
    string getName();
    void show();
}

void show() { cout << "Name : " << name << endl; }

// (a) Student (name, percent)
class Student {
    string name;
    float percent;
}

public:
    Student(string name, float percent);
    string getName();
    float getPercent();
    void show();
}

Student(string name, float percent) {
    name = name;
    percent = percent;
}

string getName() const {
    return name;
}

float getPercent() const {
    return percent;
}

void show() {
    cout << "Name : " << name << endl;
    cout << "Percent : " << percent << endl;
}

// (b) Default constructor
class StudentDefault {
    string name;
    float percent;
}

public:
    StudentDefault();
    string getName();
    float getPercent();
}

StudentDefault::StudentDefault() {
    cout << "Enter name (one word): ";
    cin >> name;
    cout << "Enter percentage: ";
    cin >> percent;
}

string getName() const {
    return name;
}

float getPercent() const {
    return percent;
}

void show() {
    cout << "Name : " << name << endl;
    cout << "Percent : " << percent << endl;
}

// (c) College (roll no., name, course)
class CollegeDefault {
    int roll;
    string name;
    string course;
}

CollegeDefault::CollegeDefault(int roll, string name, string course) {
    roll = roll;
    name = name;
    course = course;
}

void show() {
    cout << "[C1] Roll : " << roll << endl;
    cout << "Name : " << name << endl;
    cout << "Course : " << course << endl;
}

```

1. a) Department
class CollegeStudent
int roll; string name; string course;
public:
CollegeStudent(string name, string course)
int getRoll() const {return roll;}
string getName() const {return name;}
string getCourse() const {return course;}
void show()
cout << "Roll: " << roll << endl << "Name: " << name << endl << "Course: " << course << endl;

3:
1. a) copy constructor
class CollegeStudent
int roll; string name; string course;
public:
CollegeCopy (const CollegeStudent&){
roll = o.getRoll();
name = o.getName();
course = o.getCourse();

3:
void show()
cout << "Roll: " << roll << endl << "Name: " << name << endl << "Course: " << course << endl;

class ComplexOverload
int r, i;
public:
ComplexOverload (int real, int imag) : r(real), i(imag) {}
ComplexOverload (const Complex& c) : r(c.r), i(c.i) {}
void show (const string& str) {
cout << "[" << r << " + " << i << "i] " << endl;
if (str == "a") cout << "a" << endl << "a" << endl;
else cout << "b" << endl << "b" << endl;

3:
1. a) main()
1(a) sum
sumDefault a1; a2.show();
sumParam a2(a1); a3.show();
sumCopy a3(a1); a3.show();

1(b) Student
StudentDefault b1; b1.show();
StudentParam b2 ("Gauri", 91.4f); b2.show();
StudentCopy b3(b2); b3.show();

1(c) College
CollegeDefault Arg c1(101, "Gauri"); c1.show();
CollegeParam c2(102, "Anshika", "AI & DS"); c2.show();
CollegeCopy c3(c2); c3.show();

```
Macmillan  
complicated  
complicated  
complex  
complex
```

Q1
16/3

Experiment - 6

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

class Person {
protected:
    string name;
    int age;
};

class Student : protected Person {
public:
    student(string name, int age, int rollno) {
        this->name = name;
        this->age = age;
        this->rollno = rollno;
    }

    void show() {
        cout << "Name: " << name << endl;
        cout << "Age: " << age << endl;
        cout << "Roll no: " << rollno << endl;
        cout << "ID: " << id;
    }
};

int main() {
    Student St("Gaurav", 20, 10);
    St.show();
}
```


int main()
{
 Date_Car cc("TONI", 2010, "Sedan", 2000);
 cc.show();
}

Q4
include <iostream>
include <string>
using namespace std;

class Employee
{
protected:
 string name;
 int id;
};

class Manager : virtual protected: Employee
{
protected:
 string department;
public:
 Manager(string name, int id, string department) :
 this->name(name),
 this->id(id),
 this->department(department)
 {}
 void show() {
 cout << "Name: " << name << endl;
 cout << "Id: " << id << endl;
 cout << "Dept: " << department << endl;
 }
};

class Design = painted influences
painter
Story language
Author
Design Story there is like story language 235
the -> have = have
the -> it = it
the -> language = language

Very small
Centro, name is, Id is language records

Int. 1987
Francis et moi ("Céleste", 102, "IEB";
René et de ("André", 103, "Gard").
Au théâtre
de Strasbourg

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

class Department L
protected:
String Student
int staff_id

class Teacher : virtual Department
protected:
 string Subject;
 int StaffNo;

class Office : virtual protected Department &
protected:
 int fees
 String reconds;

class Student: protected Teacher, protected office 2
float type
String name
public:

Student (String Dept., String Dept., String Student, float (green))
Dept.: dept.
dfa = dept.
Subject = Subject
Name = Name

Experiment - 7

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

class Area {
public:
    float calculate(float length, float breadth);
    float calculate(float side);
};

int main()
{
    float a;
    cout << "Area : ";
    cout << a << endl;
    cout << "Area : ";
    cout << a << endl;
}

float calculate(float length, float breadth)
{
    return length * breadth;
}

float calculate(float side)
{
    return side * side;
}
```

Q2

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

class Sum {
public:
    int total(int a[], int n);
};

int total(int a[], int n)
{
    int sum = 0;
    for (int i = 0; i < n; i++)
        sum += a[i];
    return sum;
}
```

float totalAvg = 2.0; int i;

float sum;

for(i=0; i<5; i++)

sum+=

return sum;

} // main()

cout << "Sum of float" << sum << endl;

cout << "Avg of float" << sum / 5 << endl;

return 0;

3

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

```
class Teacher {
public:
```

```
    int count;
    float exp;
}
```

```
Teacher::Teacher() {
    count = 0;
    exp = 0.0;
}
```

```
Teacher::Teacher(int c, float e) {
    count = c;
    exp = e;
}
```

```
void display();
char exp;
```

```
cout << "Experience : " << exp << endl;
}
```

```
void operator=(const Teacher & t);
}
```

```
exp = t.exp;
}
```

3;

```
int main() {
    Teacher t1(0, 0);
    t1.display();
    cout << endl;
    Teacher t2(1, 100);
    t2.display();
}
```

3; // include <iostream>
using namespace std;

```
class Student {
public:
```

```
    int count;
    float exp;
}
```

```
Student::Student() {
    count = 0;
    exp = 0.0;
}
```

```
Student::Student(int c, float e) {
    count = c;
    exp = e;
}
```

```
void operator=(const Student & s);
}
```

```
void display();
}
```

```
cout << "Student Count : " << count << endl;
}
```

3;

```
int main() {
    Student s1(30, 0);
    Student s2(30, 0);
    s1.display();
    s1++;
    s1.display();
}
```

operator +

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

class Combined
{
public:
    string str;
    Combined(string s) : str(s) {}
    Combined operator+(const string &s) {
        return Combined(str + s);
    }
};

int main() {
    string s1("xyz"), s2("part"), s3;
    cout << "Concatenated String: ";
    s3 = s1 + s2;
    cout << endl;
}
```

operator <<

```
cout << str
```

3.

```
int main() {
    string s1("xyz"), s2("part"), s3;
    s3 = s1 + s2;
    cout << "Concatenated String: ";
    s3.display();
}
```

3

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

class Ilogin {
protected:
    string name, password;
public:
    virtual void accept() {
        cout << "Enter name: ";
        cin >> name;
        cout << "Enter password: ";
        cin >> password;
    }
    virtual void display() {
        cout << "Name: " << name << endl;
        cout << "password: " << password << endl;
    }
};

class Emaillogin: public Ilogin {
    string email;
public:
    void accept() {
        cout << "Email ID: ";
        cin >> email;
        Ilogin::accept();
    }
    void display() {
        cout << "\n --- Email login Details --- \n";
        cout << "Email ID: " << email << endl;
        Ilogin::display();
    }
}
```

3

```
void display() {
    cout << "\n --- Email login Details --- \n";
    cout << "Email ID: " << email << endl;
    Ilogin::display();
}
```

3

class MembershipLog : public Log {

 string memberID;

 void accept() {
 cout << "Enter Member ID: ";
 cin >> memberID;
 cout << endl;
 }

 void display() const {
 cout << "Member ID: " << memberID << endl;
 cout << "Membership Details: " << endl;
 cout << "Name: " << name;
 cout << endl;
 }

 int main() {
 MembershipLog log;
 cout << "Enter Member ID: ";
 cin >> log.memberID;
 cout << endl;
 cout << "Name: " << log.name;
 cout << endl;
 cout << "Age: " << log.age;
 cout << endl;
 cout << "Email: " << log.email;
 cout << endl;
 cout << "Phone: " << log.phone;
 cout << endl;
 cout << "Address: " << log.address;
 cout << endl;
 cout << "Gender: " << log.gender;
 cout << endl;
 cout << "Occupation: " << log.occupation;
 cout << endl;
 cout << "Display: " << log.display();
 cout << endl;
 }
 };

Experiment 4

 #include <iostream>

 #include <fstream>

 using namespace std;

 int main() {
 ifstream fin("First.txt");
 ofstream out("Second.txt");
 cout << "Enter opening file name: ";
 return 1;
 }

 char ch;
 while (!fin.eof()) {
 out.put(ch);
 }
 cout << "File Copied successfully" << endl;
 fin.close();
 out.close();
 return 0;
 }

```
class  
public  
    ~public int main()  
    {  
        ifstream file ("some.txt");  
        if(file)  
            while (file.getline (line))  
                cout << line  
        else  
            cout << "Error"  
        file.close();  
    }  
int  
{  
    char ch;  
    int digit = 0, spaces = 0;  
    while (!file.eof())  
    {  
        if (isalpha(ch))  
            digit++;  
        else if (isspace(ch))  
            spaces++;  
        ch = file.get();  
    }  
    cout << "Digit: " << digit << endl;  
    cout << "Spaces: " << spaces << endl;  
    file.close();  
}
```

```
82  
#include <iostream>  
#include <fstream>  
#include <string>  
using namespace std;  
  
int main()  
{  
    ifstream file ("file.txt");  
    if (!file) {  
        cout << "Error" << endl;  
        return 1;  
    }  
  
    string word;  
    int count = 0;  
    while (!file.eof()) {  
        cout << word << endl;  
        count++;  
    }  
    cout << "Total words: " << count << endl;  
    file.close();  
}
```

On Mar 22, 2013, at 2:27

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

int main()
{string str;
int i;
cout<<"Enter " << endl;
return 0;
}

String word targets
int counter

cout<<"Enter word to count : " << endl;
cin>>targets

```
while(!feof(word))
{if (word == targets)
    cout<<endl;
}
```

cout<<"Alphabets of " << targets << " in " << endl;
cout<<endl;

(111)

EDG072301 B
CLASSEmate

Experiment 18

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

template <typename T>
T sumArray(T arr[], int n)
{sum = 0;
for (int i = 0; i < n; i++)
 sum += arr[i];
return sum;
}

```
int main()
{int arr[5] = {1, 2, 3, 4, 5};
float flarr[5] = {1.1, 2.2, 3.3, 4.4, 5.5};
double darr[5] = {1.1, 2.2, 3.3, 4.4, 5.5};
cout<<"Sum of integer array : " << sumArray(arr, 5);
cout<<"Sum of float array : " << sumArray(flarr, 5);
cout<<"Sum of double array : " << sumArray(darr, 5);
}
```

6.2

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

```
template <typename T>
T square (T num)
{return num * num;
}
```

```
template <typename T>
String square (String (string str))
{String str2;
return str2;
}
```

ANSWER
1.A
2.A
3.O

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

```
3 void calculate()
4 {
5     #include <iostream>
6     #include <cmath>
7     using namespace std;
8
9     class Circle
10    {
11     public:
12         Circle();
13         ~Circle();
14         void calculate();
15     };
16
17     Circle::Circle()
18     {
19         cout << "Enter radius: ";
20         cin >> radius;
21     }
22
23     Circle::~Circle()
24     {
25         cout << "Area = " << pi * radius * radius;
26     }
27
28     void calculate()
29     {
30         cout << "Area = " << pi * radius * radius;
31     }
32 }
```

```
calculation.h
```

```
1 class
```

```
2 {
3     public:
```

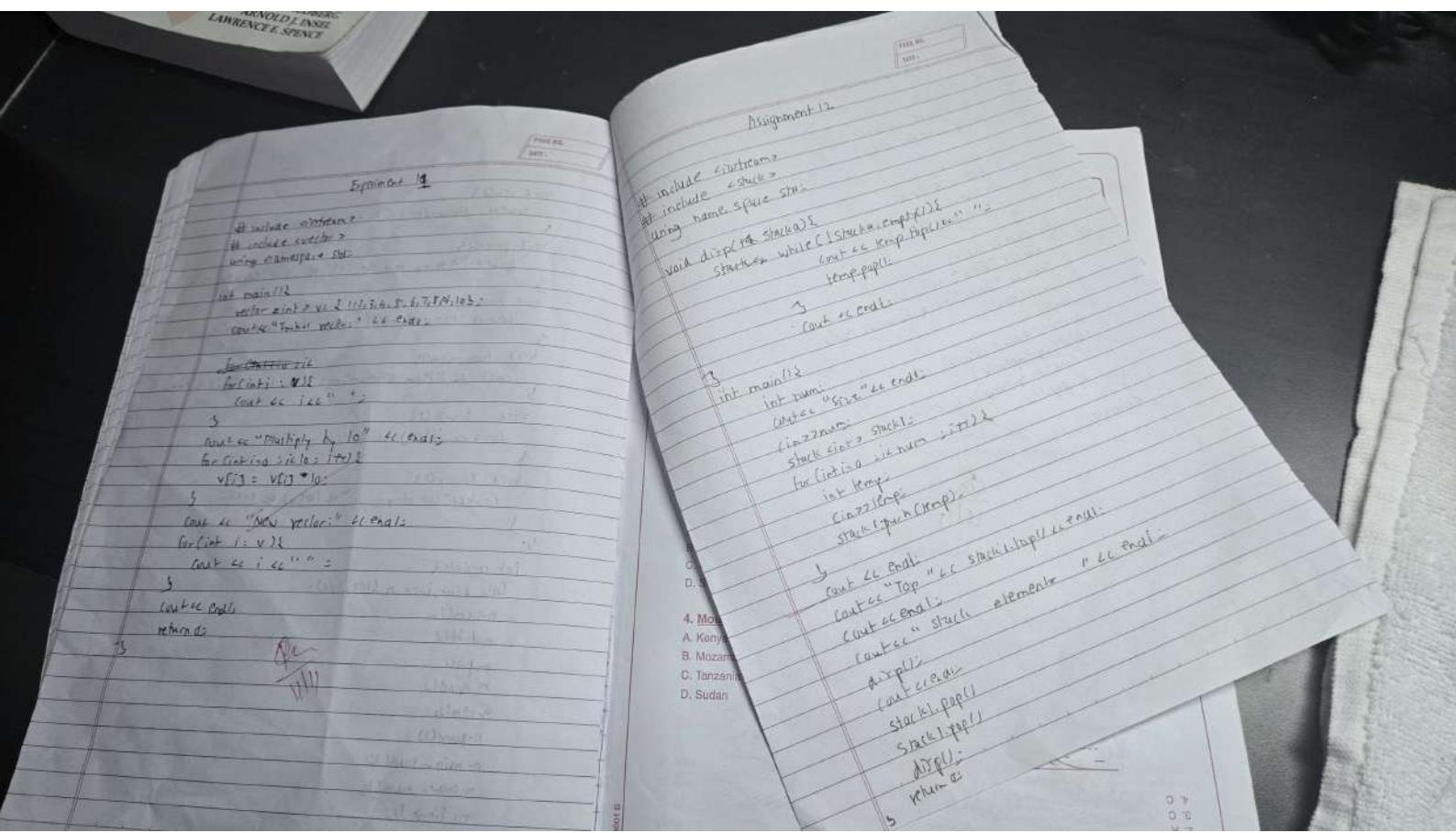
```
4     void sum()
5     {
6         cout << "Sum: " << x + y;
7     }
8 }
```

```
9 void sum()
10 {
11     cout << "Sum: " << x + y;
12 }
```

```
13 void pmt()
14 {
15     cout << "PMT: " << x * y;
16 }
```

```
17 void qud()
18 {
19     cout << "Quotient: " << x / y;
20 }
```

```
void rem()
1    cout << "Remainder: " << x % y;
2
3 void fact()
4 {
5     cout << "X! = " << x;
6     for (int i = 1; i <= x; i++)
7         cout << " * " << i;
8
9 void minmax()
10 {
11     cout << "Min of numbers: " << min(x, y);
12     cout << "Max of numbers: " << max(x, y);
13 }
14
15 void sinx()
16 {
17     cout << "Sin(x) = " << sin(x);
18 }
19
20 void cosx()
21 {
22     cout << "Cos(x) = " << cos(x);
23 }
24
25 void logx()
26 {
27     cout << "Log(x) = " << log(x);
28 }
29
30 void ln()
31 {
32     cout << "Ln(x) = " << ln(x);
33 }
34
35 void sqrt()
36 {
37     cout << "Sqrt(x) = " << sqrt(x);
38 }
39
40 void abs()
41 {
42     cout << "Abs(x) = " << abs(x);
43 }
44
45 void mod()
46 {
47     cout << "Mod(x, y) = " << x % y;
48 }
49
50 void fib()
51 {
52     cout << "Fibonacci: " << fib();
53 }
54
55 void prime()
56 {
57     cout << "Prime: " << prime();
58 }
59
60 void armstrong()
61 {
62     cout << "Armstrong: " << armstrong();
63 }
64
65 void reverse()
66 {
67     cout << "Reverse: " << reverse();
68 }
69
70 void factorial()
71 {
72     cout << "Factorial: " << factorial();
73 }
74
75 void gcd()
76 {
77     cout << "GCD: " << gcd();
78 }
79
80 void lcm()
81 {
82     cout << "LCM: " << lcm();
83 }
84
85 void power()
86 {
87     cout << "Power: " << power();
88 }
89
90 void square()
91 {
92     cout << "Square: " << square();
93 }
94
95 void cube()
96 {
97     cout << "Cube: " << cube();
98 }
99
100 void factorial()
101 {
102     cout << "Factorial: " << factorial();
103 }
104
105 void prime()
106 {
107     cout << "Prime: " << prime();
108 }
109
110 void Armstrong()
111 {
112     cout << "Armstrong: " << Armstrong();
113 }
114
115 void reverse()
116 {
117     cout << "Reverse: " << reverse();
118 }
119
120 void factorial()
121 {
122     cout << "Factorial: " << factorial();
123 }
124
125 void gcd()
126 {
127     cout << "GCD: " << gcd();
128 }
129
130 void lcm()
131 {
132     cout << "LCM: " << lcm();
133 }
134
135 void power()
136 {
137     cout << "Power: " << power();
138 }
139
140 void square()
141 {
142     cout << "Square: " << square();
143 }
144
145 void cube()
146 {
147     cout << "Cube: " << cube();
148 }
149
150 void factorial()
151 {
152     cout << "Factorial: " << factorial();
153 }
154
155 void prime()
156 {
157     cout << "Prime: " << prime();
158 }
159
160 void Armstrong()
161 {
162     cout << "Armstrong: " << Armstrong();
163 }
164
165 void reverse()
166 {
167     cout << "Reverse: " << reverse();
168 }
169
170 void factorial()
171 {
172     cout << "Factorial: " << factorial();
173 }
174
175 void gcd()
176 {
177     cout << "GCD: " << gcd();
178 }
179
180 void lcm()
181 {
182     cout << "LCM: " << lcm();
183 }
184
185 void power()
186 {
187     cout << "Power: " << power();
188 }
189
190 void square()
191 {
192     cout << "Square: " << square();
193 }
194
195 void cube()
196 {
197     cout << "Cube: " << cube();
198 }
199
200 void factorial()
201 {
202     cout << "Factorial: " << factorial();
203 }
204
205 void prime()
206 {
207     cout << "Prime: " << prime();
208 }
209
210 void Armstrong()
211 {
212     cout << "Armstrong: " << Armstrong();
213 }
214
215 void reverse()
216 {
217     cout << "Reverse: " << reverse();
218 }
219
220 void factorial()
221 {
222     cout << "Factorial: " << factorial();
223 }
224
225 void gcd()
226 {
227     cout << "GCD: " << gcd();
228 }
229
230 void lcm()
231 {
232     cout << "LCM: " << lcm();
233 }
234
235 void power()
236 {
237     cout << "Power: " << power();
238 }
239
240 void square()
241 {
242     cout << "Square: " << square();
243 }
244
245 void cube()
246 {
247     cout << "Cube: " << cube();
248 }
249
250 void factorial()
251 {
252     cout << "Factorial: " << factorial();
253 }
254
255 void prime()
256 {
257     cout << "Prime: " << prime();
258 }
259
260 void Armstrong()
261 {
262     cout << "Armstrong: " << Armstrong();
263 }
264
265 void reverse()
266 {
267     cout << "Reverse: " << reverse();
268 }
269
270 void factorial()
271 {
272     cout << "Factorial: " << factorial();
273 }
274
275 void gcd()
276 {
277     cout << "GCD: " << gcd();
278 }
279
280 void lcm()
281 {
282     cout << "LCM: " << lcm();
283 }
284
285 void power()
286 {
287     cout << "Power: " << power();
288 }
289
290 void square()
291 {
292     cout << "Square: " << square();
293 }
294
295 void cube()
296 {
297     cout << "Cube: " << cube();
298 }
299
300 void factorial()
301 {
302     cout << "Factorial: " << factorial();
303 }
304
305 void prime()
306 {
307     cout << "Prime: " << prime();
308 }
309
310 void Armstrong()
311 {
312     cout << "Armstrong: " << Armstrong();
313 }
314
315 void reverse()
316 {
317     cout << "Reverse: " << reverse();
318 }
319
320 void factorial()
321 {
322     cout << "Factorial: " << factorial();
323 }
324
325 void gcd()
326 {
327     cout << "GCD: " << gcd();
328 }
329
330 void lcm()
331 {
332     cout << "LCM: " << lcm();
333 }
334
335 void power()
336 {
337     cout << "Power: " << power();
338 }
339
340 void square()
341 {
342     cout << "Square: " << square();
343 }
344
345 void cube()
346 {
347     cout << "Cube: " << cube();
348 }
349
350 void factorial()
351 {
352     cout << "Factorial: " << factorial();
353 }
354
355 void prime()
356 {
357     cout << "Prime: " << prime();
358 }
359
360 void Armstrong()
361 {
362     cout << "Armstrong: " << Armstrong();
363 }
364
365 void reverse()
366 {
367     cout << "Reverse: " << reverse();
368 }
369
370 void factorial()
371 {
372     cout << "Factorial: " << factorial();
373 }
374
375 void gcd()
376 {
377     cout << "GCD: " << gcd();
378 }
379
380 void lcm()
381 {
382     cout << "LCM: " << lcm();
383 }
384
385 void power()
386 {
387     cout << "Power: " << power();
388 }
389
390 void square()
391 {
392     cout << "Square: " << square();
393 }
394
395 void cube()
396 {
397     cout << "Cube: " << cube();
398 }
399
400 void factorial()
401 {
402     cout << "Factorial: " << factorial();
403 }
404
405 void prime()
406 {
407     cout << "Prime: " << prime();
408 }
409
410 void Armstrong()
411 {
412     cout << "Armstrong: " << Armstrong();
413 }
414
415 void reverse()
416 {
417     cout << "Reverse: " << reverse();
418 }
419
420 void factorial()
421 {
422     cout << "Factorial: " << factorial();
423 }
424
425 void gcd()
426 {
427     cout << "GCD: " << gcd();
428 }
429
430 void lcm()
431 {
432     cout << "LCM: " << lcm();
433 }
434
435 void power()
436 {
437     cout << "Power: " << power();
438 }
439
440 void square()
441 {
442     cout << "Square: " << square();
443 }
444
445 void cube()
446 {
447     cout << "Cube: " << cube();
448 }
449
450 void factorial()
451 {
452     cout << "Factorial: " << factorial();
453 }
454
455 void prime()
456 {
457     cout << "Prime: " << prime();
458 }
459
460 void Armstrong()
461 {
462     cout << "Armstrong: " << Armstrong();
463 }
464
465 void reverse()
466 {
467     cout << "Reverse: " << reverse();
468 }
469
470 void factorial()
471 {
472     cout << "Factorial: " << factorial();
473 }
474
475 void gcd()
476 {
477     cout << "GCD: " << gcd();
478 }
479
480 void lcm()
481 {
482     cout << "LCM: " << lcm();
483 }
484
485 void power()
486 {
487     cout << "Power: " << power();
488 }
489
490 void square()
491 {
492     cout << "Square: " << square();
493 }
494
495 void cube()
496 {
497     cout << "Cube: " << cube();
498 }
499
500 void factorial()
501 {
502     cout << "Factorial: " << factorial();
503 }
504
505 void prime()
506 {
507     cout << "Prime: " << prime();
508 }
509
510 void Armstrong()
511 {
512     cout << "Armstrong: " << Armstrong();
513 }
514
515 void reverse()
516 {
517     cout << "Reverse: " << reverse();
518 }
519
520 void factorial()
521 {
522     cout << "Factorial: " << factorial();
523 }
524
525 void gcd()
526 {
527     cout << "GCD: " << gcd();
528 }
529
530 void lcm()
531 {
532     cout << "LCM: " << lcm();
533 }
534
535 void power()
536 {
537     cout << "Power: " << power();
538 }
539
540 void square()
541 {
542     cout << "Square: " << square();
543 }
544
545 void cube()
546 {
547     cout << "Cube: " << cube();
548 }
549
550 void factorial()
551 {
552     cout << "Factorial: " << factorial();
553 }
554
555 void prime()
556 {
557     cout << "Prime: " << prime();
558 }
559
560 void Armstrong()
561 {
562     cout << "Armstrong: " << Armstrong();
563 }
564
565 void reverse()
566 {
567     cout << "Reverse: " << reverse();
568 }
569
570 void factorial()
571 {
572     cout << "Factorial: " << factorial();
573 }
574
575 void gcd()
576 {
577     cout << "GCD: " << gcd();
578 }
579
580 void lcm()
581 {
582     cout << "LCM: " << lcm();
583 }
584
585 void power()
586 {
587     cout << "Power: " << power();
588 }
589
590 void square()
591 {
592     cout << "Square: " << square();
593 }
594
595 void cube()
596 {
597     cout << "Cube: " << cube();
598 }
599
600 void factorial()
601 {
602     cout << "Factorial: " << factorial();
603 }
604
605 void prime()
606 {
607     cout << "Prime: " << prime();
608 }
609
610 void Armstrong()
611 {
612     cout << "Armstrong: " << Armstrong();
613 }
614
615 void reverse()
616 {
617     cout << "Reverse: " << reverse();
618 }
619
620 void factorial()
621 {
622     cout << "Factorial: " << factorial();
623 }
624
625 void gcd()
626 {
627     cout << "GCD: " << gcd();
628 }
629
630 void lcm()
631 {
632     cout << "LCM: " << lcm();
633 }
634
635 void power()
636 {
637     cout << "Power: " << power();
638 }
639
640 void square()
641 {
642     cout << "Square: " << square();
643 }
644
645 void cube()
646 {
647     cout << "Cube: " << cube();
648 }
649
650 void factorial()
651 {
652     cout << "Factorial: " << factorial();
653 }
654
655 void prime()
656 {
657     cout << "Prime: " << prime();
658 }
659
660 void Armstrong()
661 {
662     cout << "Armstrong: " << Armstrong();
663 }
664
665 void reverse()
666 {
667     cout << "Reverse: " << reverse();
668 }
669
670 void factorial()
671 {
672     cout << "Factorial: " << factorial();
673 }
674
675 void gcd()
676 {
677     cout << "GCD: " << gcd();
678 }
679
680 void lcm()
681 {
682     cout << "LCM: " << lcm();
683 }
684
685 void power()
686 {
687     cout << "Power: " << power();
688 }
689
690 void square()
691 {
692     cout << "Square: " << square();
693 }
694
695 void cube()
696 {
697     cout << "Cube: " << cube();
698 }
699
700 void factorial()
701 {
702     cout << "Factorial: " << factorial();
703 }
704
705 void prime()
706 {
707     cout << "Prime: " << prime();
708 }
709
710 void Armstrong()
711 {
712     cout << "Armstrong: " << Armstrong();
713 }
714
715 void reverse()
716 {
717     cout << "Reverse: " << reverse();
718 }
719
720 void factorial()
721 {
722     cout << "Factorial: " << factorial();
723 }
724
725 void gcd()
726 {
727     cout << "GCD: " << gcd();
728 }
729
730 void lcm()
731 {
732     cout << "LCM: " << lcm();
733 }
734
735 void power()
736 {
737     cout << "Power: " << power();
738 }
739
740 void square()
741 {
742     cout << "Square: " << square();
743 }
744
745 void cube()
746 {
747     cout << "Cube: " << cube();
748 }
749
750 void factorial()
751 {
752     cout << "Factorial: " << factorial();
753 }
754
755 void prime()
756 {
757     cout << "Prime: " << prime();
758 }
759
760 void Armstrong()
761 {
762     cout << "Armstrong: " << Armstrong();
763 }
764
765 void reverse()
766 {
767     cout << "Reverse: " << reverse();
768 }
769
770 void factorial()
771 {
772     cout << "Factorial: " << factorial();
773 }
774
775 void gcd()
776 {
777     cout << "GCD: " << gcd();
778 }
779
780 void lcm()
781 {
782     cout << "LCM: " << lcm();
783 }
784
785 void power()
786 {
787     cout << "Power: " << power();
788 }
789
790 void square()
791 {
792     cout << "Square: " << square();
793 }
794
795 void cube()
796 {
797     cout << "Cube: " << cube();
798 }
799
800 void factorial()
801 {
802     cout << "Factorial: " << factorial();
803 }
804
805 void prime()
806 {
807     cout << "Prime: " << prime();
808 }
809
810 void Armstrong()
811 {
812     cout << "Armstrong: " << Armstrong();
813 }
814
815 void reverse()
816 {
817     cout << "Reverse: " << reverse();
818 }
819
820 void factorial()
821 {
822     cout << "Factorial: " << factorial();
823 }
824
825 void gcd()
826 {
827     cout << "GCD: " << gcd();
828 }
829
830 void lcm()
831 {
832     cout << "LCM: " << lcm();
833 }
834
835 void power()
836 {
837     cout << "Power: " << power();
838 }
839
840 void square()
841 {
842     cout << "Square: " << square();
843 }
844
845 void cube()
846 {
847     cout << "Cube: " << cube();
848 }
849
850 void factorial()
851 {
852     cout << "Factorial: " << factorial();
853 }
854
855 void prime()
856 {
857     cout << "Prime: " << prime();
858 }
859
860 void Armstrong()
861 {
862     cout << "Armstrong: " << Armstrong();
863 }
864
865 void reverse()
866 {
867     cout << "Reverse: " << reverse();
868 }
869
870 void factorial()
871 {
872     cout << "Factorial: " << factorial();
873 }
874
875 void gcd()
876 {
877     cout << "GCD: " << gcd();
878 }
879
880 void lcm()
881 {
882     cout << "LCM: " << lcm();
883 }
884
885 void power()
886 {
887     cout << "Power: " << power();
888 }
889
890 void square()
891 {
892     cout << "Square: " << square();
893 }
894
895 void cube()
896 {
897     cout << "Cube: " << cube();
898 }
899
900 void factorial()
901 {
902     cout << "Factorial: " << factorial();
903 }
904
905 void prime()
906 {
907     cout << "Prime: " << prime();
908 }
909
910 void Armstrong()
911 {
912     cout << "Armstrong: " << Armstrong();
913 }
914
915 void reverse()
916 {
917     cout << "Reverse: " << reverse();
918 }
919
920 void factorial()
921 {
922     cout << "Factorial: " << factorial();
923 }
924
925 void gcd()
926 {
927     cout << "GCD: " << gcd();
928 }
929
930 void lcm()
931 {
932     cout << "LCM: " << lcm();
933 }
934
935 void power()
936 {
937     cout << "Power: " << power();
938 }
939
940 void square()
941 {
942     cout << "Square: " << square();
943 }
944
945 void cube()
946 {
947     cout << "Cube: " << cube();
948 }
949
950 void factorial()
951 {
952     cout << "Factorial: " << factorial();
953 }
954
955 void prime()
956 {
957     cout << "Prime: " << prime();
958 }
959
960 void Armstrong()
961 {
962     cout << "Armstrong: " << Armstrong();
963 }
964
965 void reverse()
966 {
967     cout << "Reverse: " << reverse();
968 }
969
970 void factorial()
971 {
972     cout << "Factorial: " << factorial();
973 }
974
975 void gcd()
976 {
977     cout << "GCD: " << gcd();
978 }
979
980 void lcm()
981 {
982     cout << "LCM: " << lcm();
983 }
984
985 void power()
986 {
987     cout << "Power: " << power();
988 }
989
990 void square()
991 {
992     cout << "Square: " << square();
993 }
994
995 void cube()
996 {
997     cout << "Cube: " << cube();
998 }
999
999 void factorial()
1000 {
1001     cout << "Factorial: " << factorial();
1002 }
```



PAGE NO.
DATE:

2) #include <iostream>
#include <queue>
using namespace std;

int main() {
 queue<int> q;
 for (int i = 0; i <= 10; i++) {
 q.push(i * 10);

}

cout << endl;
cout << "Front" << q.front() << endl;
cout << "Back" << q.back() << endl;
cout << endl;

q.pop();
cout << "Queue elements:" << endl;
while (!q.empty()) {

cout << q.front() << " "

q.pop();

}

return 0;

Ques
1/1/1