|  |
| --- |
|  |
| OOPS Experiments |
|  |
|  |
| **Vansh Sukhija**  **12112021** |
|  |

|  |
| --- |
|  |

Experiment 1(i)-

#include<iostream>

using *namespace* std;

*struct* phone{

*int* areacode;

*int* exchange;

*int* number;

};

*int* main(){

    phone p1, p2;

    p1.areacode = 212, p1.exchange = 767, p1.number = 8900;

    cout<<"Enter area code: "; cin>>p2.areacode;

    cout<<"Enter exchange: "; cin>>p2.exchange;

    cout<<"Enter number: "; cin>>p2.number;

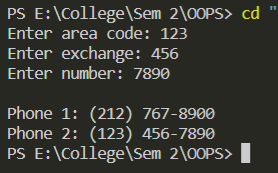
    cout<<endl<<"Phone 1: ("<<p1.areacode<<") "<<p1.exchange<<'-'<<p1.number;

    cout<<endl<<"Phone 2: ("<<p2.areacode<<") "<<p2.exchange<<'-'<<p2.number;

    return 0;

}

Output-



Experiment 1(ii)-

#include<iostream>

using *namespace* std;

*struct* point{

*int* x, y;

};

*int* main(){

    point p1, p2, p3;

    p1.x = 1, p1.y = 2;

    p2.x = 3, p2.y = 4;

    p3.x = p1.x + p2.x;

    p3.y = p1.y + p2.y;

    cout<<"Point 1: ("<<p1.x<<", "<<p1.y<<")"<<endl;

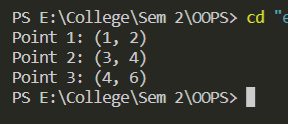
    cout<<"Point 2: ("<<p2.x<<", "<<p2.y<<")"<<endl;

    cout<<"Point 3: ("<<p3.x<<", "<<p3.y<<")"<<endl;

    return 0;

}

Output-



Experiment 1(iii)-

#include<iostream>

using *namespace* std;

*struct* employee{

*private:*

    string name;

*int* id;

*int* salary;

    string address;

*public:*

*void* getinput(){

        cout<<"Enter Name: ";

        getline(cin, name);

        cout<<"Enter ID: ";

        cin>>id;

        cout<<"Enter Salary: ";

        cin>>salary;

        fflush(stdin);

        cout<<"Enter address: ";

        getline(cin, address);

    }

*void* printoutput(){

        cout<<endl<<name<<endl<<id<<endl<<salary<<endl<<address<<endl;

    }

};

*int* main(){

    employee emp;

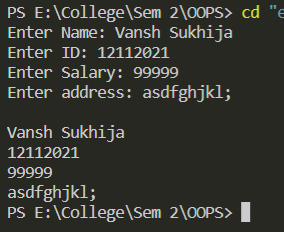
    emp.getinput();

    emp.printoutput();

    return 0;

}

Output-



Experiment 2(i)-

#include<iostream>

using *namespace* std;

*class* Employee{

*private:*

    string name;

*int* id;

*int* salary;

    string address;

*public:*

*void* getinput(){

        cout<<"Enter Name: ";

        getline(cin, name);

        cout<<"Enter ID: ";

        cin>>id;

        cout<<"Enter Salary: ";

        cin>>salary;

        fflush(stdin);

        cout<<"Enter address: ";

        getline(cin, address);

    }

*void* printoutput(){

        cout<<endl<<name<<endl<<id<<endl<<salary<<endl<<address<<endl;

    }

};

*int* main(){

    Employee emp;

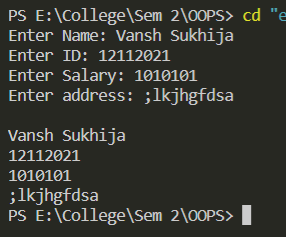
    emp.getinput();

    emp.printoutput();

    return 0;

}

Output-



Experiment 2(ii)-

#include<iostream>

using *namespace* std;

*class* INT{

*private:*

*int* variable;

*public:*

    INT(*int* *var*=0){

        variable = *var*;

    }

*void* init(*int* *var*=0){

        variable = *var*;

    }

*void* display(){

        cout<<variable<<endl;

    }

    INT add(INT *a*, INT *b*){

        variable = *a*.variable + *b*.variable;

    }

};

*int* main(){

*int* num1, num2;

    cout<<"Enter 2 numbers: ";

    cin>>num1>>num2;

    INT a(num1), b, c(num2);

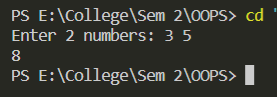
    b.add(a, c);

    b.display();

    return 0;

}

Output-



Experiment 2(iii)-

#include<iostream>

using *namespace* std;

*class* bank{

*private:*

    string name;

*long* *long* acc\_no;

    string acc\_type;

*long* *long* amount;

*public:*

*void* init(string *n*, *long* *long* *num*, string *type*, *long* *long* *bal*=0){

        name = *n*;

        acc\_no = *num*;

        acc\_type = *type*;

        amount = *bal*;

    }

*void* deposit(*int* *dep*){

        amount += *dep*;

        cout<<*dep*<<" deposited"<<endl<<"Current balance: "<<amount<<endl;

    }

*void* withdraw(*int* *draw*){

        if(*draw*>amount)

            cout<<"Unsufficient balance"<<endl;

        else{

            amount -= *draw*;

            cout<<*draw*<<" withdrawn"<<endl<<"Current balance: "<<amount<<endl;

        }

    }

*void* display(){

        cout<<"Name: "<<name<<endl;

        cout<<"Balance: "<<amount<<endl;

    }

};

*int* main(){

    bank bk;

    string name;

*long* *long* acc\_no;

    string acc\_type;

*long* *long* amount;

    cout<<"Enter name: ";

    getline(cin, name);

    fflush(stdin);

    cout<<"Enter account number: ";

    cin>>acc\_no;

    fflush(stdin);

    cout<<"Enter account type: ";

    getline(cin, acc\_type);

    cout<<"Enter current balance: ";

    cin>>amount;

    bk.init(name, acc\_no, acc\_type, amount);

    cout<<endl;

    bk.deposit(100);

    cout<<endl;

    bk.withdraw(40);

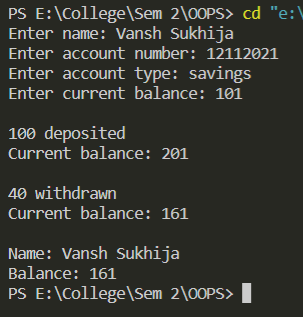
    cout<<endl;

    bk.display();

    return 0;

}

Output-



Experiment 2(iv)-

#include<iostream>

#include<conio.h>

using *namespace* std;

*class* TollBooth{

*private:*

*unsigned* *int* cars=0;

*double* money=0;

*public:*

*void* paycar(){

        cars++, money+=0.5;

    }

*void* nopaycar(){

        cars++;

    }

*void* display(){

        cout<<endl<<" Total Cars: "<<cars<<endl<<"Total Money: "<<money<<endl;

    }

};

*int* main(){

    TollBooth tb;

*char* input;

    cout<<"Enter 'y' for PayCar\nEnter 'n' for NoPayCar\nEnter 'Esc' to exit\n";

    do{

        input = getche();

        if(input=='y')

            tb.paycar();

        else if(input=='n')

            tb.nopaycar();

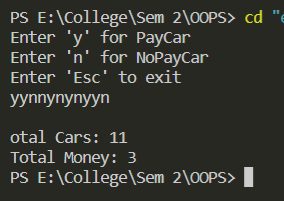
    }while(input!=27);

    tb.display();

    return 0;

}

Output-



Experiment 3(i)-

#include<iostream>

using *namespace* std;

*class* Employee{

*private:*

    string name;

*long* number;

*public:*

*void* getdata(*int* *i*){

        fflush(stdin);

        cout<<"Enter name of Employee "<<*i*+1<<": ";

        getline(cin, name);

        cout<<"Enter Employee number: ";

        cin>>number;

    }

*void* putdata(){

        cout<<name<<endl<<number<<endl;

    }

};

*int* main(){

*int* n;

    cout<<"Enter number of employees: ";

    cin>>n;

    Employee emp[n];

    for(*int* i=0; i<n; i++){

        emp[i].getdata(i);

        fflush(stdin);

    }

    cout<<endl<<"Employee Data"<<endl;

    for(*int* i=0; i<n; i++){

        emp[i].putdata();

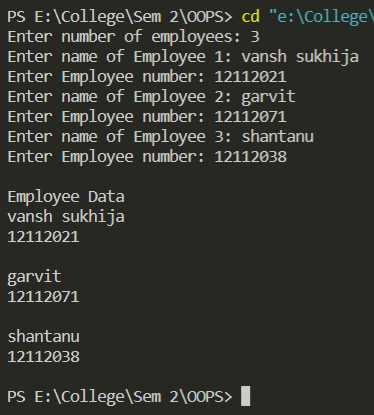
        cout<<endl;

    }

    return 0;

}

Output-



Experiment 3(ii)-

#include<iostream>

using *namespace* std;

typedef *class* student{

*private:*

*int* roll\_no;

    string name;

    string address;

    string city;

*int* pin;

*int* sem;

*int* rank;

    string branch;

*public:*

*void* set(*int* *i*){

        cout<<"Enter Roll no. of student "<<*i*+1<<": ";

        cin>>roll\_no;

        fflush(stdin);

        cout<<"Enter name: ";

        getline(cin, name);

        fflush(stdin);

        cout<<"Enter Address: ";

        getline(cin, address);

        fflush(stdin);

        cout<<"Enter city: ";

        getline(cin, city);

        fflush(stdin);

        cout<<"Enter pin: ";

        cin>>pin;

        fflush(stdin);

        cout<<"Enter sem: ";

        cin>>sem;

        fflush(stdin);

        cout<<"Enter rank: ";

        cin>>rank;

        fflush(stdin);

        cout<<"Enter branch: ";

        getline(cin, branch);

        fflush(stdin);

    }

*void* get(){

        cout<<endl<<name<<endl<<roll\_no<<endl<<address<<endl<<city<<endl<<pin<<endl<<sem<<endl<<rank<<endl<<branch<<endl;

    }

    string getbranch(){

        return branch;

    }

*int* getsem(){

        return sem;

    }

}stud;

*int* main(){

*int* n;

    cout<<"Enter number of students: ";

    cin>>n;

    stud st[n];

    for(*int* i=0; i<n; i++){

        st[i].set(i);

        fflush(stdin);

    }

    string branch;

*int* sem;

    cout<<"Enter branch to display data: ";

    cin>>branch;

    cout<<"Enter semester to display data: ";

    cin>>sem;

    for(*int* i=0; i<n; i++){

        if(st[i].getbranch()==branch && st[i].getsem()==sem)

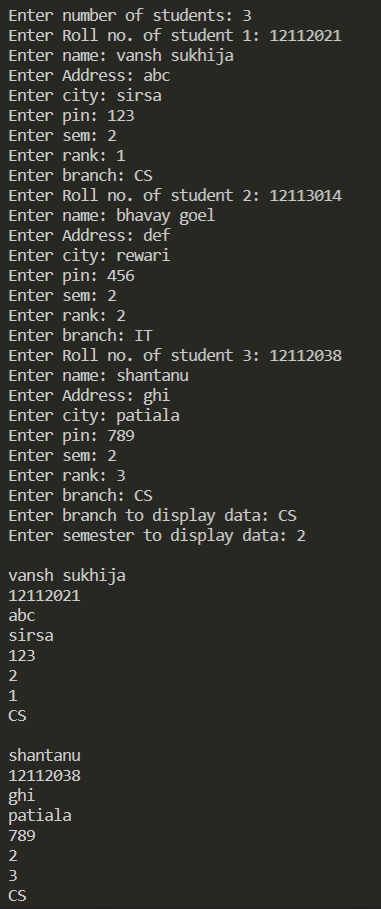
            st[i].get();

    }

    return 0;

}

Output-



Experiment 3(iii)-

#include<iostream>

using *namespace* std;

*class* Distance

{

*private:*

*int* feet;

*float* inches;

*public:*

*void* getDist(){

        cin>>feet>>inches;

    }

*void* showDist(){

        cout<<feet<<"' "<<inches<<endl;

    }

*float* distadd(){

        return feet\*12 + inches;

    }

};

*int* main(){

*int* n;

*float* avg=0;

    cout<<"Enter number of distances: ";

    cin>>n;

    Distance dt[n];

    for(*int* i=0; i<n; i++){

        dt[i].getDist();

        avg += dt[i].distadd()/n;

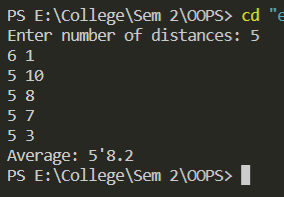
    }

    cout<<"Average: "<<(*int*)avg/12<<"'"<<avg - (*int*)avg/12\*12;

    return 0;

}

Output-



Experiment 4(ii)-

#include <iostream>

using *namespace* std;

*class* fraction

{

*private:*

*int* num;

*int* den;

*public:*

    fraction(*int* *a*, *int* *b*)

    {

        num = *a*;

        den = *b*;

    }

*int* geti()

    {

        return (num);

    }

*int* getf()

    {

        return (den);

    }

*void* seti(*int* *a*)

    {

        num = *a*;

    }

*void* setf(*int* *a*)

    {

        den = *a*;

    }

*void* operator-=(fraction *f2*)

    {

        num = num - *f2*.num;

    }

};

*int* lcm(*int* *a*, *int* *b*)

{

    if (*a* % *b* == 0)

        return (*a*);

    else

    {

        return (*a* \* *b*);

    }

}

*int* main()

{

*int* h, i;

    fraction f1(1, 5), f2(2, 7);

    if (f1.getf() != f2.getf())

    {

        h = f1.getf();

        i = f2.getf();

        if (f1.getf() > f2.getf())

        {

            f1.setf(lcm(f1.getf(), f2.getf()));

        }

        else

        {

            f1.setf(lcm(f2.getf(), f1.getf()));

        }

        f2.seti(f2.geti() \* (f1.getf() / i));

        f1.seti(f1.geti() \* (f1.getf() / h));

        f1 -= f2;

    }

    else

        f1 -= f2;

    if (f1.geti() == 0)

    {

        printf("diff= 0");

    }

    else

        printf("diff= %d/%d", f1.geti(), f1.getf());

}

Output-



4(iii)-

#include <iostream>

using *namespace* std;

*class* fraction

{

*private:*

*int* num;

*int* den;

*public:*

    fraction(*int* *a*, *int* *b*)

    {

        num = *a*;

        den = *b*;

    }

*int* geti()

    {

        return (num);

    }

*int* getf()

    {

        return (den);

    }

*void* seti(*int* *a*)

    {

        num = *a*;

    }

*void* setf(*int* *a*)

    {

        den = *a*;

    }

*void* operator\*=(fraction *f2*)

    {

        num = num \* *f2*.num;

        den = den \* *f2*.den;

    }

};

*int* main()

{

*int* h, i;

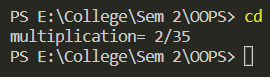
    fraction f1(1, 5), f2(2, 7);

    f1 \*= f2;

    printf("multiplication= %d/%d", f1.geti(), f1.getf());

}

Output-



4(iv)-

#include <iostream>

using *namespace* std;

*class* complex

{

*private:*

*float* real;

*float* img;

*public:*

*float* getr()

    {

        return (real);

    }

*float* geti()

    {

        return (img);

    }

    complex(*float* *a* = 0, *float* *b* = 0)

    {

        real = *a*;

        img = *b*;

    }

    complex operator+(complex *f3*)

    {

        complex temp;

        temp.real = real + *f3*.real;

        temp.img = img + *f3*.img;

        return (temp);

    }

    complex operator-(complex *f3*)

    {

        complex temp;

        temp.real = real - *f3*.real;

        temp.img = img - *f3*.img;

        return (temp);

    }

    complex operator\*(complex *f3*)

    {

        complex temp;

        temp.real = real \* *f3*.real;

        temp.img = img \* *f3*.img;

        return (temp);

    }

    complex operator/(complex *f3*)

    {

        complex temp;

        temp.real = real / *f3*.real;

        temp.img = img / *f3*.img;

        return (temp);

    }

};

*int* main()

{

    complex f1;

    complex f2(2, 3), f3(5, 4);

    f1 = f2 + f3;

    printf("numbers are=\n");

    printf("%.0f + %0.0fi\n", f2.getr(), f2.geti());

    printf("%.0f + %0.0fi\n", f3.getr(), f3.geti());

    printf("addition = %0.f + %0.fi\n", f1.getr(), f1.geti());

    f1 = f2 - f3;

    if (f1.geti() < 0)

        printf("subtraction = %0.f - %0.fi\n", f1.getr(), -f1.geti());

    else

        printf("subtraction = %0.f + %0.fi\n", f1.getr(), f1.geti());

    f1 = f2 \* f3;

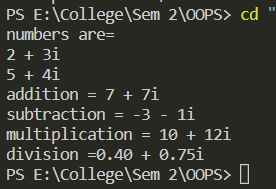
    printf("multiplication = %.0f + %0.fi\n", f1.getr(), f1.geti());

    f1 = f2 / f3;

    printf("division =%.2f + %0.2fi\n", f1.getr(), f1.geti());

}

Output-



4(v)-

#include <iostream>

using *namespace* std;

*class* compare2;

*class* compare1

{

*private:*

*int* a;

*public:*

    compare1(*int* *b* = 0)

    {

        a = *b*;

    }

    friend *void* compare(compare1 *c1*, compare2 *c2*);

};

*class* compare2

{

*private:*

*int* d;

*public:*

    compare2(*int* *b* = 0)

    {

        d = *b*;

    }

    friend *void* compare(compare1 *c1*, compare2 *c2*);

};

*void* compare(compare1 *c1*, compare2 *c2*)

{

    if (*c1*.a > *c2*.d)

        printf("elememt of class 1>class 2");

    else if (*c2*.d > *c1*.a)

        printf("elememt of class 1<class 2");

    else

        printf("elememt of class 1=class 2");

}

*int* main()

{

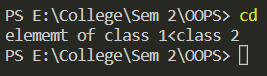
    compare1 c(2);

    compare2 c1(4);

    compare(c, c1);

}

Output-



4(vi)-

#include <iostream>

using *namespace* std;

*class* fraction

{

*private:*

*int* num;

*int* den;

*public:*

    fraction(*int* *a*, *int* *b*)

    {

        num = *a*;

        den = *b*;

    }

*int* geti()

    {

        return (num);

    }

*int* getf()

    {

        return (den);

    }

*void* seti(*int* *a*)

    {

        num = *a*;

    }

*void* setf(*int* *a*)

    {

        den = *a*;

    }

*void* operator\*=(fraction *f2*)

    {

        num = num \* *f2*.den;

        den = den \* *f2*.num;

    }

};

*int* main()

{

*int* h, i;

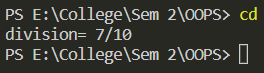
    fraction f1(1, 5), f2(2, 7);

    f1 \*= f2;

    printf("division= %d/%d", f1.geti(), f1.getf());

}

Output-



4(vii)-

#include <iostream>

using *namespace* std;

*class* employee

{

*private:*

    string name;

*long* *int* number;

*int* salary;

*public:*

*void* getdata(string *a*, *long* *int* *b*, *int* *c*)

    {

        name = *a*;

        number = *b*;

        salary = *c*;

    }

    string getn()

    {

        return (name);

    }

*long* *int* getno()

    {

        return (number);

    }

*int* gets()

    {

        return (salary);

    }

*void* putdata()

    {

        cout << name << "\t" << number << "\t" << salary << "\n";

    }

};

*void* sorting(*class* employee *e*[], *int* *n*)

{

*class* employee t;

*int* i, j;

    for (i = 0; i < *n* - 1; i++)

    {

        for (j = 0; j < *n* - 1 - i; j++)

        {

            if (*e*[j].gets() > *e*[j + 1].gets())

            {

                t = *e*[j];

*e*[j] = *e*[j + 1];

*e*[j + 1] = t;

            }

        }

    }

}

*int* main()

{

*int* n, x, c;

    string a;

*long* *int* b;

    cout << "Enter the number of employee=";

    cin >> n;

    employee e[n];

    for (x = 0; x < n; x++)

    {

        cout << "enter the name,number ,salary of employee" << x + 1 << "=" << endl;

        cin >> a >> b >> c;

        e[x].getdata(a, b, c);

    }

    sorting(e, n);

    cout << "NAME\t NUMBER\t\tSALARY\n";

    for (x = 0; x < n; x++)

    {

        e[x].putdata();

    }

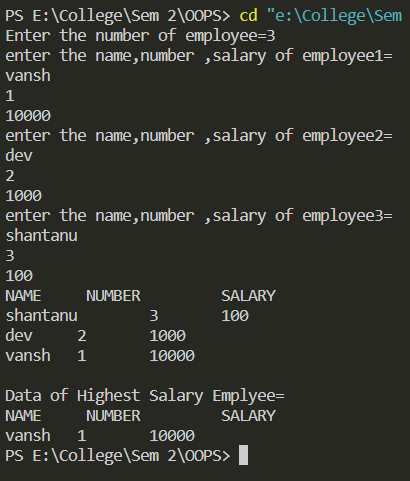
    printf("\nData of Highest Salary Emplyee=\n");

    cout << "NAME\t NUMBER\t\tSALARY\n";

    e[n - 1].putdata();

}

Output-



Experiment 5(i)-

#include<iostream>

using *namespace* std;

*class* stack{

*private:*

*int* size, top, \*ptr;

*public:*

    stack(*int* *s*){

        size = *s*;

        this->ptr = (*int* \*)malloc(size\*sizeof(*int*));

        top = -1;

    }

*void* push(*int* *num*){

        if(top == size-1)

            cout<<"Stack Overflow"<<endl;

        else{

            top++;

            ptr[top] = *num*;

        }

    }

*int* pop(){

*int* num;

        if(top==-1)

            cout<<"Stack Underflown"<<endl;

        else{

            num = ptr[top];

            top--;

        }

        return num;

    }

*int* stackfull(){

        if(top == size-1)

            return 1;

        else

            return 0;

    }

*int* stackempty(){

        if(top == -1)

            return 1;

        else

            return 0;

    }

*void* display(){

        cout<<"Stack: ";

        for(*int* i=0; i<=top; i++){

            cout<<ptr[i]<<' ';

        }

        cout<<endl;

    }

};

*int* main(){

    stack st(5);

    st.push(1);

    st.display();

    st.push(2);

    st.display();

    st.push(3);

    st.display();

    st.push(4);

    st.display();

    st.pop();

    st.display();

    st.pop();

    st.display();

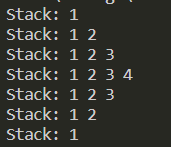
    st.pop();

    st.display();

    return 0;

}

Output-



5(ii)-

#include <iostream>

#include <cstdlib>

using *namespace* std;

*class* Queue{

*private:*

*int* front;

*int* rear;

*int* data[100];

*public:*

*int* size;

    Queue(){

        front = -1;

        rear = -1;

    }

*void* insertqueue(){

        if(front == -1 && rear == -1){

            front++, rear++;

            cout<<"Enter data: ";

            cin>>data[rear];

        }

        else if(rear < size - 1){

            rear++;

            cout<<"Enter data: ";

            cin>>data[rear];

        }

        else if(rear > size - 2){

            cout<<"Queue Overflown"<<endl;

        }

    }

*void* deletequeue(){

        if (front == -1 && rear == -1)

            cout<<"Queue is Empty"<<endl;

        else if (front < rear){

            cout<<"Deleted item is: "<<data[front]<<endl;

            front++;

        }

    }

*void* display(){

        cout<<"Queue: ";

        for (*int* i=front; i<=rear; i++){

            cout<<data[i]<<" ";

        }

        cout<<endl;

    }

};

*int* main(){

*int* n, m, ch;

    Queue s;

    cout << "Enter the size of Queue: ";

    cin >> s.size;

    while(1){

        cout<<"Enter 1 for insert"<<endl;

        cout<<"Enter 2 for delete"<<endl;

        cout<<"Enter 3 for display"<<endl;

        cout<<"Enter 4 to exit"<<endl;

        cout<<"Enter here: ";

        cin>>ch;

        cout<<endl;

        switch (ch){

            case 1:

                s.insertqueue();

                break;

            case 2:

                s.deletequeue();

                break;

            case 3:

                s.display();

                break;

            case 4:

                exit(100);

        }

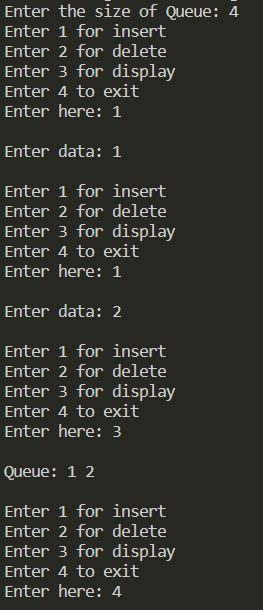
        cout<<endl;

    }

    return 0;

}

Output-



5(iii)-

#include <iostream>

using *namespace* std;

*class* array

{

*private:*

*int* arr;

*public:*

    array()

    {

        arr = 0;

    };

*void* set(*int* *a*)

    {

        arr = *a*;

    }

*int* get()

    {

        return (arr);

    }

    ~array(){

        // cout<<"descructor invoked";

    };

};

*int* binary\_search(*class* array *arr*[], *int* *h*, *int* *key*)

{

*int* l = 1, mid;

    while (l <= *h*)

    {

        mid = (l + *h*) / 2;

        if (*key* == *arr*[mid].get())

            return (mid);

        if (*key* < *arr*[mid].get())

*h* = mid - 1;

        else

            l = mid + 1;

    }

    return (0);

}

*void* bubble(*class* array *arr*[], *int* *n*)

{

*int* i, j, temp;

    for (i = 0; i < *n* - 1; i++)

    {

        for (j = 0; j < *n* - 1 - i; j++)

        {

            if (*arr*[j].get() > *arr*[j + 1].get())

            {

                temp = *arr*[j].get();

*arr*[j].set(*arr*[j + 1].get());

*arr*[j + 1].set(temp);

            }

        }

    }

}

*int* main()

{

*int* x, a, key, j;

    array arr[10];

    printf("enter the element of sorted array elements=\n");

    for (x = 0; x < 10; x++)

    {

        scanf("%d", &a);

        arr[x].set(a);

    }

    printf("enter the value of key=");

    scanf("%d", &key);

    j = binary\_search(arr, 10, key);

    if (j != 0)

        printf("found the element");

    else

        printf("not found the element");

    printf("\nenter the element of array elements=\n");

    for (x = 0; x < 10; x++)

    {

        scanf("%d", &a);

        arr[x].set(a);

    }

    bubble(arr, 10);

    printf("sorted array=\n");

    for (x = 0; x < 10; x++)

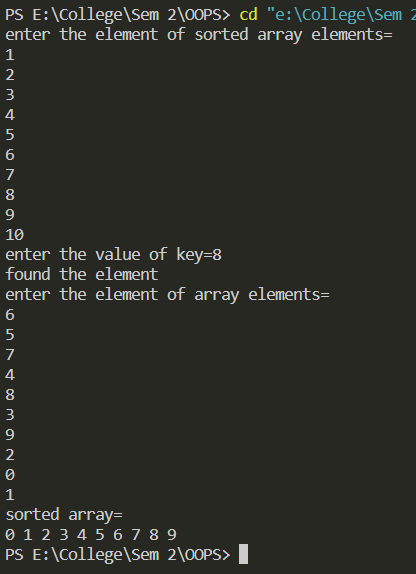
    {

        printf("%d ", arr[x].get());

    }

}

Output-



Experiment 6(i)-

#include <iostream>

using *namespace* std;

*class* loc

{

*int* x;

*public:*

*void* getData()

    {

        cout << "Enter Distance: ";

        cin >> x;

    }

    loc operator-(loc *dis*)

    {

        loc temp;

        temp.x = x - *dis*.x;

        return temp;

    }

*bool* operator<(loc *dis*)

    {

        if (x < *dis*.x)

        {

            return true;

        }

        return false;

    }

*void* display()

    {

        cout << x << endl;

    }

};

*int* main()

{

    loc dis1, dis2;

    dis1.getData();

    dis2.getData();

    if (dis1 < dis2)

    {

        cout << "Difference between the two distances is negative" << endl;

        return 0;

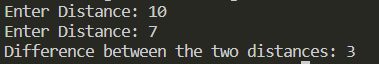
    }

    cout << "Difference between the two distances: ";

    (dis1 - dis2).display();

}

Output-



6(ii)-

#include <iostream>

using *namespace* std;

*class* time

{

*private:*

*int* hrs;

*int* min;

*int* sec;

*public:*

*void* getData()

    {

        cout << "Enter Time: ";

        cin >> hrs;

        cin >> min;

        cin >> sec;

    }

    time operator+(time *t*)

    {

        time temp;

        temp.hrs = hrs + t.hrs;

        temp.min = min + t.min;

        temp.sec = sec + t.sec;

        if (temp.sec >= 60)

        {

            temp.sec = temp.sec - 60;

            temp.min = temp.min + 1;

        }

        if (temp.min >= 60)

        {

            temp.min = temp.min - 60;

            temp.hrs = temp.hrs + 1;

        }

        return temp;

    }

*void* display()

    {

        cout << "Time: " << hrs << ":" << min << ":" << sec << endl;

    }

};

*int* main()

{

    time t1, t2;

    t1.getData();

    t1.display();

    t2.getData();

    t2.display();

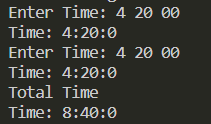
    cout << "Total Time" << endl;

    (t1 + t2).display();

    return 0;

}

Output-



6(iii)-

#include <iostream>

using *namespace* std;

*class* frac

{

*float* x;

*public:*

    frac() {}

*void* getData()

    {

        cout << "Enter value: ";

        cin >> x;

    }

*void* display()

    {

        cout << "Entered value: ";

        cout << x << endl;

    }

    friend *bool* operator>(frac *f1*, frac *f2*);

};

*bool* operator>(frac *f1*, frac *f2*)

{

    if (*f1*.x > *f2*.x)

    {

        return true;

    }

    return false;

}

*int* main()

{

    frac f1, f2;

    f1.getData();

    f2.getData();

    cout << endl;

    f1.display();

    f2.display();

    if (f1 > f2)

    {

        cout << "Fraction 1 is greater than Fraction 2";

        return 0;

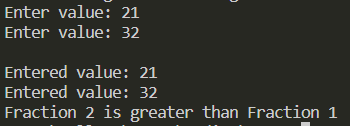
    }

    cout << "Fraction 2 is greater than Fraction 1";

    return 0;

}

Output-



6(iv)-

#include <iostream>

using *namespace* std;

*class* frac

{

*float* x;

*public:*

    frac() {}

*void* getData()

    {

        cout << "Enter value: ";

        cin >> x;

    }

*void* display()

    {

        cout << "Entered value: ";

        cout << x << endl;

    }

    friend *bool* operator&&(frac *f1*, frac *f2*);

};

*bool* operator&&(frac *f1*, frac *f2*)

{

    return *f1*.x && *f2*.x;

}

*int* main()

{

    frac f1, f2;

    f1.getData();

    f2.getData();

    cout << endl;

    f1.display();

    f2.display();

    cout<<endl;

    if ((f1 && f2) == 0)

    {

        cout << "One of the fraction is zero"<<endl;

        return 0;

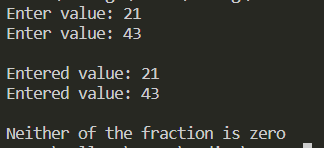
    }

    cout << "Neither of the fraction is zero"<<endl;

    return 0;

}

Output-



6(v)-

#include <iostream>

using *namespace* std;

*class* frac

{

*float* x;

*public:*

    frac() {}

*void* getData()

    {

        cout << "Enter value: ";

        cin >> x;

    }

*void* display()

    {

        cout << "Entered value: ";

        cout << x << endl;

    }

*int* operator()()

    {

        x = (*int*)x;

        return x;

    }

};

*int* main(){

    frac f;

    f.getData();

    cout << endl;

    f.display();

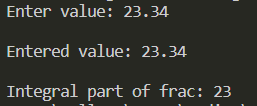
    cout << endl;

    cout << "Integral part of frac: " << f() << endl;

    return 0;

}

Output-



6(vi)-

#include <iostream>

using *namespace* std;

*class* frac

{

*float* x;

*public:*

    frac() {}

*void* getData()

    {

        cout << "Enter value: ";

        cin >> x;

    }

*void* display()

    {

        cout << "Entered value: ";

        cout << x << endl;

    }

    frac operator[](frac *f*)

    {

        frac temp;

        temp.x = *f*.x - (*int*)*f*.x;

        cout << temp.x << endl;

        return temp;

    }

};

*int* main()

{

    frac f1, f;

    f1.getData();

    cout << endl;

    f1.display();

    cout << endl;

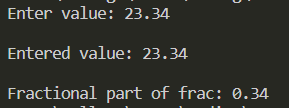
    cout << "Fractional part of frac: ";

    f = f[f1];

    return 0;

}

Output-



Experiment 7(i)-

#include<iostream>

using *namespace* std;

*class* student{

*private:*

    string name;

*char* gender;

*int* rollno, marks, age;

*public:*

*void* init(){

        fflush(stdin);

        cout<<"Enter Name: ";

        cin>>name;

        fflush(stdin);

        cout<<"Enter gender: ";

        cin>>gender;

        cout<<"Enter Roll No: ";

        cin>>rollno;

        cout<<"Enter Total Marks: ";

        cin>>marks;

        cout<<"Enter Age: ";

        cin>>age;

    }

*void* display(){

        cout<<name<<endl<<gender<<endl<<rollno<<endl<<marks<<endl<<age<<endl<<endl;

    }

};

*int* main(){

*int* n;

    cout<<"Enter number of students: ";

    cin>>n;

    student st[n], \*ptr;

    ptr = &st[0];

    for(*int* i=0; i<n; i++){

        (ptr+i)->init();

        cout<<endl;

    }

    cout<<"-----------Displaying Data-----------"<<endl;

    for(*int* i=0; i<n; i++){

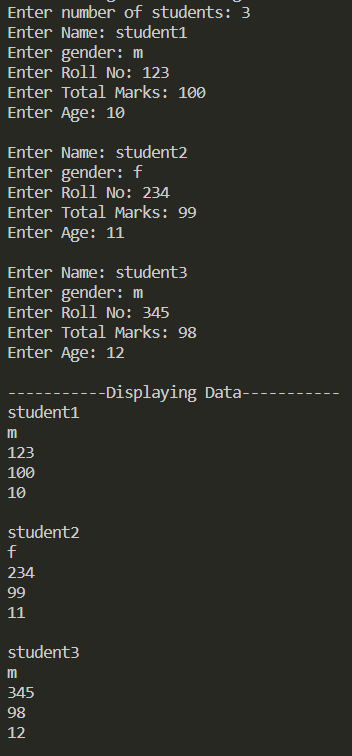
        (ptr+i)->display();

    }

    return 0;

}

Output-



7(ii)-

#include <iostream>

#include <string.h>

using *namespace* std;

*class* String{

*private:*

*char* \*name;

*int* length;

*public:*

    String(){

        length = 0;

        name = new *char*[length + 1];

    }

    String(*char* \**s*){

        length = strlen(*s*);

        name = new *char*[length + 1];

        strcpy(name, *s*);

    }

*void* upit(){

        for (*int* i=0; i<length; i++){

            name[i] = name[i] + 'A' - 'a';

        }

    }

*void* display(){

        cout << "Name: " << name << endl;

    }

};

*int* main(){

*char* name[20];

    cout<<"Enter Name: ";

    cin>>name;

    String s(name);

    s.display();

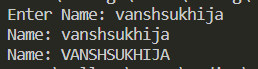
    s.upit();

    s.display();

    return 0;

}

Output-



7(iii)-

#include <iostream>

using *namespace* std;

*class* Array{

*private:*

*int* l,i;

*int* \*arr;

*void* extend(){

*int* \*arr2;

        if(i == 0){

            arr = new *int*;

            i++;

        }

        else{

            arr2 = new *int*[++i];

            for (*int* j = 0; j <= i - 2; j++){

                arr2[j] = arr[j];

            }

            delete (arr);

            arr = arr2;

        }

    }

*void* contract(){

*int* \*arr2;

        arr2 = new *int*[--i];

        for (*int* j = 0; j < i; j++){

            arr2[j] = arr[j];

        }

        delete (arr);

        arr = arr2;

    }

*public:*

    Array(*int* *x*){

        arr = NULL;

        l = *x*;

        i = 0;

    }

    Array(Array &*a*){

        l = *a*.l;

        arr = *a*.arr;

    }

    ~Array(){

        delete (arr);

    }

*int* append(*int* *x*){

        if(i != l){

            extend();

            arr[i - 1] = *x*;

            return 1;

        }

        else

            return 0;

    }

*int* chop(){

        if(i != 0){

            contract();

            return 1;

        }

        else{

            return 0;

        }

    }

*void* show(){

        for(*int* j = 0; j < i; j++){

            cout << arr[j] << ' ';

        }

    }

};

*int* main(){

*int* i, a, j;

    Array x(5);

    while(j){

        cout<<"Select from the following"<<endl;

        cout<<"1. Append"<<endl;

        cout<<"2. Chop"<<endl;

        cin>>i;

        if(i == 1){

            cout<<"Enter the value: ";

            cin>>a;

            x.append(a);

        }

        if(i == 2)

            x.chop();

        cout<<"Want more operation?(0/1): ";

        cin>>j;

    }

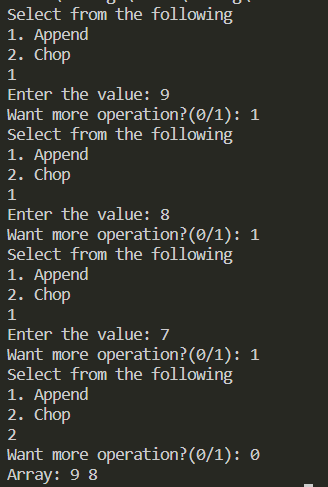
    cout<<"Array: ";

    x.show();

    return 0;

}

Output-



Experiment 8(i)-

#include <iostream>

using *namespace* std;

*class* student

{

*protected:*

*int* roll;

    string name;

*public:*

    student() {}

*void* getData1()

    {

        cout<<"Enter Roll No: ";

        cin >> roll;

        cout<<"Enter Name: ";

        cin >> name;

    }

*void* display1()

    {

        cout << "Roll No: " << roll << endl;

        cout << "Name: " << name << endl;

    }

};

*class* exam : *public* student

{

*protected:*

*float* phy;

*float* chem;

*float* math;

*float* ss;

*float* eng;

*float* hindi;

*public:*

    exam() {}

*void* getData2()

    {

        cout << "Enter phy marks: ";

        cin >> phy;

        cout << "Enter chem marks: ";

        cin >> chem;

        cout << "Enter math marks: ";

        cin >> math;

        cout << "Enter ss marks: ";

        cin >> ss;

        cout << "Enter eng marks: ";

        cin >> eng;

        cout << "Enter hindi marks: ";

        cin >> hindi;

    }

};

*class* result : *public* exam

{

*float* totalMarks;

*public:*

*int* total()

    {

        totalMarks = phy + chem + math + ss + eng + hindi;

        return totalMarks;

    }

*void* display2()

    {

        cout << "Total Marks: " << totalMarks << endl;

    }

};

*int* main()

{

*int* n;

    cout << "Enter no of students: ";

    cin >> n;

    result st[n];

    cout << "Enter student details " << endl;

    cout<<endl;

    for (*int* i = 0; i < n; i++)

    {

        st[i].getData1();

        st[i].getData2();

        cout<<endl;

    }

    for (*int* i = 0; i < n; i++)

    {

        st[i].total();

    }

    cout << endl;

    cout << "Exam Result " << endl;

    cout << endl;

    for (*int* i = 0; i < n; i++)

    {

        st[i].display1();

        st[i].display2();

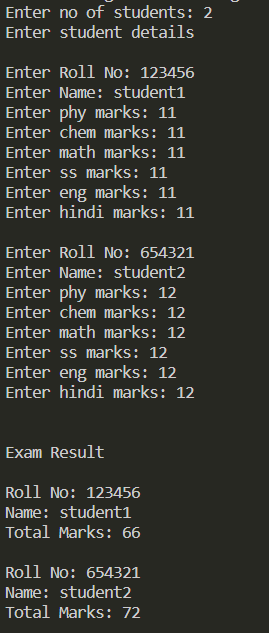
        cout<<endl;

    }

    return 0;

}

Output-



8(ii)-

#include <iostream>

using *namespace* std;

*class* publication

{

    string title;

*float* price;

*public:*

    publication(){}

*void* getData1()

    {

        cout << "Enter Title: ";

        cin >> title;

        cout << "Enter Price: ";

        cin >> price;

    }

*void* putData1()

    {

        cout << "Title of Book: " << title << endl;

        cout << "Price of Book: " << price << endl;

    }

};

*class* book : *public* publication

{

*int* page;

*public:*

    book() {}

*void* getData2()

    {

        cout << "Enter Page count: ";

        cin >> page;

    }

*void* putData2()

    {

        cout << "Page count: " << page << endl;

    }

};

*class* tape : *public* publication

{

*float* playingtime;

*public:*

    tape() {}

*void* getData3()

    {

        cout << "Enter playing time(in min): ";

        cin >> playingtime;

    }

*void* putData3()

    {

        cout << "Playing Time: " << playingtime << endl;

    }

};

*int* main()

{

*int* n;

    cout << "Enter no of books: ";

    cin >> n;

*int* m;

    cout << "Enter no of tapes: ";

    cin >> m;

    book b[n];

    tape t[m];

    cout << "Input Data: " << endl;

    cout << endl;

    for (*int* i = 0; i < n; i++)

    {

        cout << "Enter Book Details" << endl;

        b[i].getData1();

        b[i].getData2();

        cout << endl;

    }

    for (*int* i = 0; i < m; i++)

    {

        cout << "Enter Tape Details" << endl;

        t[i].getData1();

        t[i].getData3();

        cout << endl;

    }

    cout << "Displaying Data: " << endl;

    cout << endl;

    for (*int* i = 0; i < n; i++)

    {

        b[i].putData1();

        b[i].putData2();

        cout << endl;

    }

    for (*int* i = 0; i < m; i++)

    {

        t[i].putData1();

        t[i].putData3();

        cout << endl;

    }

    return 0;

}

Output-

