Programming Practices Using C++.

-supervised by:

Prof. Kaushik Majumdar, and

Prof. Mihir Singh

Rohit Das

B. Tech(Computer Sc. and Engg)

Roll: 30000114022

Regn. No.:143000110023

5th Semester,2016

Maulana Abul Kalam Azad University of Technology,

West Bengal.

LaTeX 2016

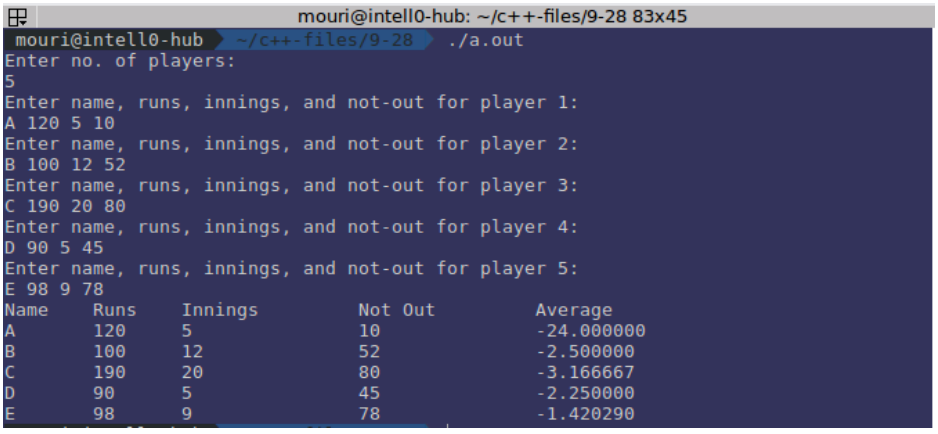
INDEX

1. Write a program for a cricket player having the following attributes: *name, *runs, *innings and *not out.
The program should calculate the average of each player, and show the same as output.

Program:

```
#include <iostream>
#include<cstdio>
using namespace std;
typedef struct player{
    char name[50];
    int runs;
    int innings;
    int not_out;
    float avg;
}player;
int main() {
        printf("Enter no. of players:\n");
        int n;
        scanf("%d",&n);
        player a[n];
        for(int i=0;i<n;i++){
            printf("Enter name, runs, innings, and not-out for player %d:
             \n",(i+1));
            scanf("%s",a[i].name);
            scanf("%d %d%d",&a[i].runs,&a[i].innings,&a[i].not_out);
        }
        for(int i=0;i<n;i++){
            a[i].avg=1.0*a[i].runs/(a[i].innings-a[i].not_out);
        }
        printf("Name\tRuns\tInnings\t\tNot Out\t\tAverage \n");
        for(int i=0;i<n;i++){
            printf("%s\t%d\t%d\t\t%d\t\t%f \n",a[i].name,a[i].runs,
            a[i].innings,a[i].not_out,a[i].avg);
        }
        return 0;
}
```

Output:

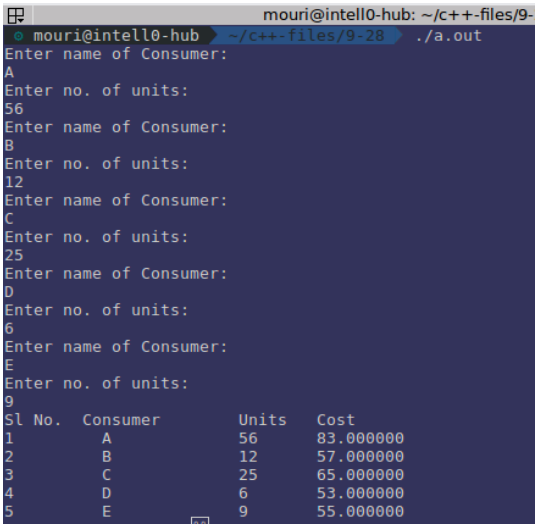


2. Write a program for a consumer having the following attributes: name and no. of units bought. The program should calculate the total cost for each consumer, and show the same as output.

Program:

```
#include <iostream>
#include <cstdio>
using namespace std;
typedef struct consumer{
    int sl;
    char name[50];
    int units;
    float cost;
}consumer;
int costing(int n){
    float cost1=50;
    if(n<=100)cost1+=n*0.60;
    else if(n>100&&n<=300)cost1+=((n-100)*0.80)+60;
    else cost1+=140+(n-300)*0.90;
    if(cost1>300)cost1=cost1-(cost1*0.15);
    return cost1;
}
int main() {
    int count,i=0;
    count=1;
    consumer a[50];
        while(1){
            a[i].sl=i+1;
            cout<<"Enter name of Consumer:"<<endl;
            scanf("%s",a[i].name);
            cout<<"Enter no. of units:"<<endl;
            cin>>a[i].units;
            count++;
            if(count>5)break;
            i++;
        }
        i=0;
        cout<<"Sl No.\t"<<"Consumer\t"<<"Units\t"<<"Cost"<<endl;
        while(i<count){
            a[i].cost=costing(a[i].units);
            if(a[i].units!=-1&&a[i].units!=0)printf("%d  \t  %s\t\t%d\t
            %f\n",a[i].sl,a[i].name,
            a[i].units,a[i].cost);
            i++;
        }
        return 0;
}
```

Output:



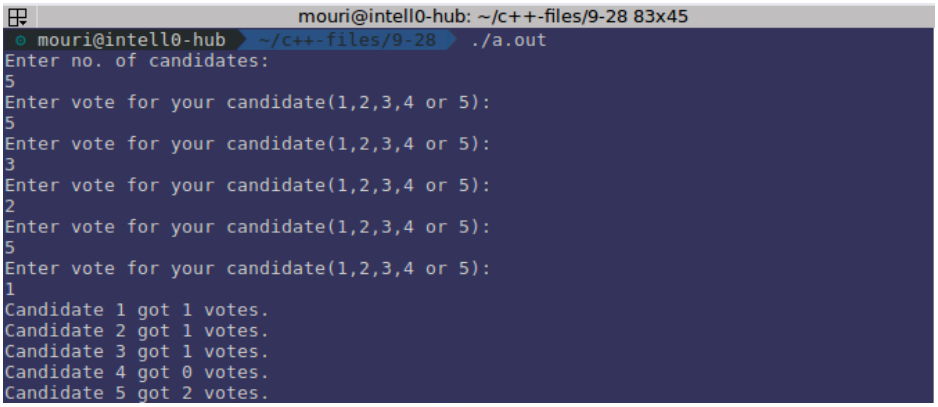
.

3. Write a program taking as input the no. of candidates for an election and the votes each got. The output will display how many votes each got.

Program:

```
#include <iostream>
using namespace std;
int main() {
        int n,vote;
        static int arr[100];
        int i=0;
        cout<<"Enter no. of candidates:"<<endl;
        cin>>n;
        i=0;
        while(i<n){
            cout<<"Enter vote for your candidate(1,2,3,4 or 5):"<<endl;
            cin>>vote;
            arr[vote]++;
            i++;
            if(vote==-1||vote>5)break;
        }
        i=0;
        while(i<n){
            cout<<"Candidate "<<i+1<<" got "<<arr[i+1]<<" votes."<<endl;
            i++;
        }
        return 0;
}
```

Output:



.

4. Write a program with a function to take array(s) and size as input and:
***return largest and smallest element, and**
***search an element and return 1 if present, 0 if absent.**

Program:

```
#include<iostream>
#include<cstdio>
#define swap(a,b) (a)=(b)-(a)+((b)=(a))
using namespace std;
int largest(int a[],int size){
        int largest=a[0];
        for(int i=1;i<size;i++)if(a[i]>largest)largest=a[i];
        return largest;
}
int smallest(int a[],int size){
        int smallest=a[0];
        for(int i=1;i<size;i++)if(a[i]<smallest)smallest=a[i];
        return smallest;
}
int search(int a[],int size,int key){
        int flag=0;
        for(int i=0;i<size;i++){if(key==a[i]){flag=1; break; }}
        if(flag==1)return 1;
        else return 0;
}
int main(){
        int n;
        printf("Enter a size for array: \n");
        scanf("%d",&n);
        int a[n];
        printf("Enter elements for array: \n");
        for(int i=0;i<n;i++)scanf("%d",&a[i]);
        for(int i=0;i<n;i++)printf("%d ",a[i]);
        printf("\n");
        printf("Largest element is %d \n",largest(a,n));
        printf("Smallest element is %d \n",smallest(a,n));
        int key=0;
```

```
        printf("Enter key to be searched:\n");
        scanf("%d",&key);
        if(search(a,n,key)==1)printf("Element %d was found in array.\n",
        key);
        else printf("Element %d was not found in array.\n",key);
}
```

Output:

```
mouri@intell0-hub ~/c++-files/10-19 ./a.out
Enter a size for array:
5
Enter elements for array:
6 4 8 9 -1
6 4 8 9 -1
Largest element is 9
Smallest element is -1
Enter key to be searched:
5
Element 5 was not found in array.
```

.

5. Write a program to calculate factorial of a number using function.

Program:

```
#include<iostream>
#include<cstdio>
using namespace std;
int fact(int d){
        int f=1;
        while(d)f*=d--;
        return f;
}
int main(){
        printf("Enter no. to calculate factorial:\n");
        int n=0;
        scanf("%d",&n);
        printf("The factorial of %d is %d \n",n,fact(n));
}
```

Output:

```
mouri@intell0-hub ~/c++-files/10-19 ./a.out
Enter no. to calculate factorial:
5
The factorial of 5 is 120
```

.

6. Write a program to find out the sum of the series using function: $1 + 22 + 32 + 42 + ...$till n terms (taking n as parameter) and return sum of the series.

Program:

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

```
        int s=12,add=0;
        int count=0;
        while(count<n){
                s+=((count++)*10);
                add+=s;
        }
        return add-11;
}
int main(){
        int n;
        printf("Enter no of terms: \n");
        scanf("%d",&n);
        printf("The sum of series is %d.\n",sum(n));
}
```

Output:

```
mouri@intell0-hub ~/c++-files/10-19 ./a.out
Enter no of terms:
5
The sum of series is 249.
```

.

7. Write a program to pass Complex numbers as objects and add them.

Program:

```
#include<iostream>
#include<cstdio>
using namespace std;
class Complex{
        public:
                int x,y;
                Complex(){}
                Complex(int x,int y)
                {
                        this->x=x;
                        this->y=y;
                }
                static Complex sum(Complex a,Complex b)
                {
                        Complex c;
                        c.x=a.x+b.x;
                        c.y=a.y+b.y;
                        return c;
                }
                void display()
                {
                        printf("THe number is %d+%di.\n",x,y);
                }
};
int main(){
        Complex c1(10,20);
        c1.display();
```

```
        Complex c2(20,30);
        c2.display();
        printf("After addition:\n");
        Complex c3;
        c3=Complex::sum(c1,c2);
        c3.display();
}
```

Output:

```
mouri@intell0-hub ~/c++-files/11-9 ./a.out
THe number is 10+20i.
THe number is 20+30i.
After addition:
THe number is 30+50i.
```

.

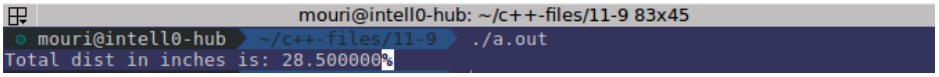
8. Write a program to take distances as input, one in inches, other in metres and to convert the distances in inches and add.

Program:

```
#include<iostream>
#include<cstdio>
using namespace std;
class DM{
        int a,b;
        public:
                DM(){}
                DM(int a,int b){
                        this->a=a;
                        this->b=b;
                }
                friend float convert(DM);
};
class DB{
        float x,y;
        public:
                DB(){}
                DB(float x,float y){
                        this->x=x;
                        this->y=y;
                }
                friend float convert(DM obj1){
                        return obj1.a*2.25f;
                }
                void add(DM a){
                        y+=convert(a);
                        printf("Total dist in inches is: %f",y);
                }
};
int main(){
        DM a(10,0);
        DB b(0,6);
```

```
        b.add(a);
}
```

Output:



.

9. Write a program to create String class having one uninitialized and a parameterized constructor.

Program:

```
#include<iostream>
#include<cstdio>
#include<cstring>
using namespace std;
class String{
        public:
                string s;
                String(){       s="";   }
                String(string str){     s=str;          }
                void concat(string s2){ s=s+s2; }
                void display(){ cout<<s<<endl;  }
};
int main(){
        String s2("Hello ");
        s2.concat("World");
        s2.display();
}
```

Output:

.

10. Write a program to create a student class and create 10 objects and display the roll no. of the students. Use static data member and array to create objects. Do not use constructors.

Program:

```
#include<iostream>
#include<cstdio>
using namespace std;
class Student{
        int roll;
        public:
                static int count;
                Student(){}
                void init(){    roll=++count;           }
```

```
                void display(){ printf("%d ",roll);     }
};
int Student::count;
int main(){
        Student arr[10];
        for(int i=0;i<10;i++){
                int x=Student::count;
                arr[i].init();
        }
        printf("The rolls are: \n");
        for(int i=0;i<10;i++){
                printf("Student %d: ",i);
                arr[i].display();
                printf("\n");
        }
        return 0;
}
```

Output:

```
mouri@intell0-hub ~/c++-files/11-9 ./a.out
The rolls are:
Student 0: 1
Student 1: 2
Student 2: 3
Student 3: 4
Student 4: 5
Student 5: 6
Student 6: 7
Student 7: 8
Student 8: 9
Student 9: 10
```

.

11. Write a program to values of objects using 'friend' functions.

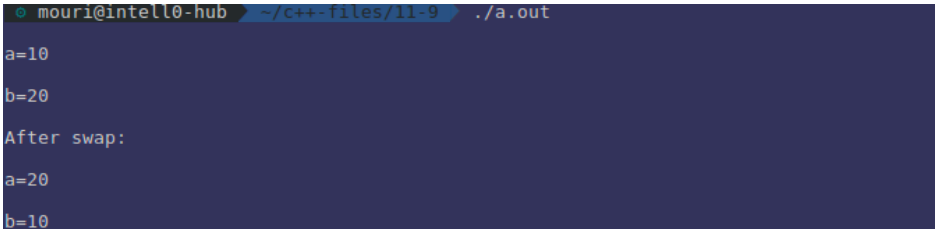
Program:

```
#include<iostream>
#include<cstdio>
using namespace std;
class Swap
{
        int n;
        public:
                Swap(){}
                Swap(int n){    this->n=n;      }
                void show(){    printf("%d \n",n);      }
                friend void swap(Swap,Swap);
};
class Swap2{
        public:
                friend void swap(Swap a,Swap b){
                        Swap temp;
                        printf("\na="); a.show();
                        printf("\nb="); b.show();
                        printf("\nAfter swap:\n");
                        temp.n=a.n;  a.n=b.n;  b.n=temp.n;
```

```
                        printf("\na="); a.show();
                        printf("\nb="); b.show();
                }

};
int main(){
        Swap a(10);     Swap b(20);
        Swap2 s;        swap(a,b);
}
```

Output:



.

12. Write a program to create a class Matrix of size m x n. Define all possible matrix operations for Matrix type objects.

Program:

```
#include<iostream>
#include<cstdio>
using namespace std;
class Matrix{
        int r,c;
        public:
        int a[100][100];
        Matrix(){}
        Matrix(int row,int col){        r=row;c=col;    }
        Matrix operator+(Matrix);
        Matrix operator-(Matrix);
        Matrix operator*(Matrix);
        void input(void);
        void display(void);
};
Matrix Matrix::operator+(Matrix c){
        Matrix temp(2,2);
        for(int i=0;i<c.r;i++){
                for(int j=0;j<c.c;j++){
                        temp.a[i][j]=a[i][j]+c.a[i][j];
                }
        }
        return (temp);
}
Matrix Matrix::operator-(Matrix c){
        Matrix temp(2,2);
        for(int i=0;i<c.r;i++){
```

```
                for(int j=0;j<c.c;j++){
                        temp.a[i][j]=a[i][j]-c.a[i][j];
                }
        }
        return (temp);
}
Matrix Matrix::operator*(Matrix c){
        Matrix temp(2,2);
        for(int i=0;i<r;i++){
                for(int j=0;j<c.c;j++){
                        for(int k=0;k<c.r;k++){
                                temp.a[i][j]=a[i][k]*c.a[k][j];
                        }
                }
        }
        return (temp);
}
void Matrix::input(void){
        printf("Enter data for matrix(%d x %d):",r,c);
        for(int i=0;i<r;i++){
                for(int j=0;j<c;j++){
                        scanf("%d",&a[i][j]);
                }
        }
}
void Matrix::display(void){
        printf("The data for matrix \n");
        for(int i=0;i<r;i++){
                for(int j=0;j<c;j++)printf("%d ",a[i][j]);
                printf("\n");
        }
}
int main(){
        Matrix m1,m2,m3;
        m1=Matrix(2,2); m1.input();
        m2=Matrix(2,2); m2.input();
        m3=m1+m2;
        printf("Matrix 1=");    m1.display();
        printf("Matrix 2=");    m2.display();
        cout<<"Sum is:"<<endl;
        printf("Matrix 3=");    m3.display();
        m3=m1-m2;
        cout<<"DIfference is:"<<endl;
        printf("Matrix 3=");    m3.display();
        m3=m1*m2;
        cout<<"Product is:"<<endl;
        printf("Matrix 3=");    m3.display();
        return 0;
}
```

Output:

```
mouri@intell0-hub ~/c++-files/11-16 ./a.out
Enter data for matrix(2 x 2):1 2 3 4
Enter data for matrix(2 x 2):5 6 7 8
Matrix 1=The data for matrix
1 2
3 4
Matrix 2=The data for matrix
5 6
7 8
Sum is:
Matrix 3=The data for matrix
6 8
10 12
DIfference is:
Matrix 3=The data for matrix
-4 -4
-4 -4
Product is:
Matrix 3=The data for matrix
14 16
28 32
```

.

13. Write a program to create a class FLOAT that contains one float data member. Overload all the four arithmetic operators so that they operate on the objects of FLOAT.

Program:

```
#include<iostream>
#include<cstdio>
using namespace std;
class Float{
        float x;
        public:
                Float(){}
                Float(float a){ x=a;    }
                Float operator+(Float);
                Float operator-(Float);
                Float operator*(Float);
                Float operator/(Float);
                void display(void);
};
Float Float::operator+(Float c){        return Float(x+c.x);    }
Float Float::operator-(Float c){        return Float(x-c.x);    }
Float Float::operator*(Float c){        return Float(x*c.x);    }
Float Float::operator/(Float c){        return Float(x/c.x);    }
void Float::display(void){      printf("%f \n",x);      }
int main(){
        Float f1,f2,f3;
        f1=Float(1.5);  f2=Float(2.5);
        f3=f1+f2;
        printf("f1="); f1.display();
        printf("f2="); f2.display();
        cout<<"Sum is:"<<endl;
        printf("f3="); f3.display();
        cout<<"Difference is:"<<endl;
        f3=f1-f2;
        printf("f3="); f3.display();
        cout<<"Product is:"<<endl;
        f3=f1*f2;
```

```
        printf("f3="); f3.display();
        cout<<"Quotient is:"<<endl;
        f3=f1/f2;
        printf("f3="); f3.display();
        return 0;
}
```

Output:

```
mouri@intel0-hub ~/c++-files/11-16 ./a.out
f1=1.500000
f2=2.500000
Sum is:
f3=4.000000
Difference is:
f3=-1.000000
Product is:
f3=3.750000
Quotient is:
f3=0.600000
```

.

14. Write a program to compare two strings by overloading the '==' operator.

Program:

```
#include<iostream>
#include<cstdio>
#include<string>
using namespace std;
class String{
        string s;
        public:
        String(){       s="";   }
        String(string _s){      s=_s;   }
        string operator==(String);
};
string String::operator==(String str){
        if(s==str.s)return "Equals\n";
        else return "Not equals\n";
}
int main(){
        String s1("Hello"),s2("Hola"),s3("Hello");
        cout<<"Comparing Hello and Hola:"<<endl;
        cout<<(s1==s2);
        cout<<"Comparing Hello and Hello:"<<endl;
        cout<<(s1==s3);
}
```

Output:

```
mouri@intel0-hub ~/c++-files/11-16 ./a.out
Comparing Hello and Hola:
Not equals
Comparing Hello and Hello:
Equals
```

.

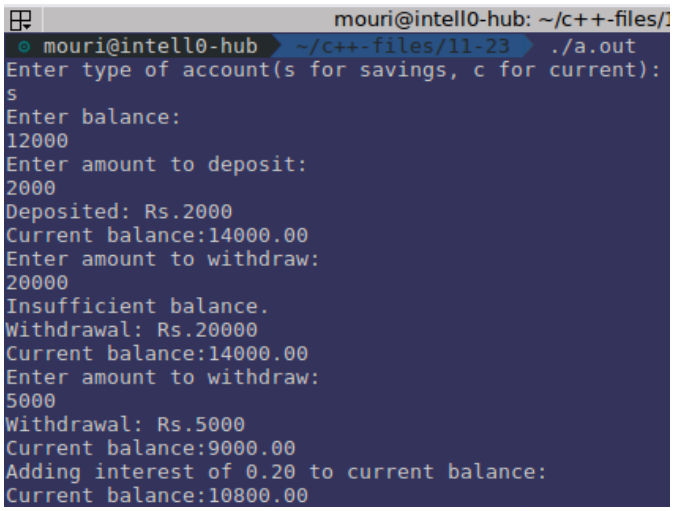
15. Write a program to create a class account that stores customer name, account number and type of account. Create two more classes for current a/c and savings a/c. The current a/c will have:
∗cheque facility,
∗ minimum balance and deduction for balance below that,
∗deposit and withdrawal. The saving a/c will have similar member methods, except for cheque and minimum balance, it will have an interest calculation. Do not use constructors.

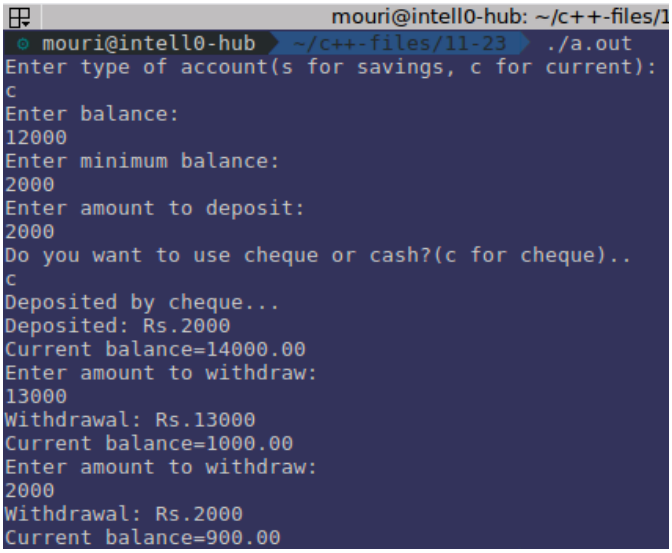
Program:

```
#include<iostream>
#include<cstdio>
#include<string>
#include<stdlib.h>
#include<cmath>
#define ll long long int
using namespace std;
class Account{
    string customer_name,acct_type;
    ll acct;
};
class Cur_acct:public Account{
    double balance,minBal;
    public:
        void setbal(double a,double b)  {
            balance=a;
            minBal=b;
        }
        void withdrawal(int n)  {
            if(balance<minBal)balance-=(balance*0.10);
            else if(n>balance)printf("Insufficient balance.\n");
            else balance-=n;
        }
        void deposit(int n){    balance+=n;     }
        void cheque(){ cout<<"Deposited by cheque..."<<endl;    }
        void show(){
            printf("Current balance=%.2f\n",balance);
        }
};
class Sav_acct:public Account{
    double balance;
    public:
        void setbal(double a){    balance=a;    }
        void withdrawal(int n){
            if(n>balance)printf("Insufficient balance.\n");
            else balance-=n;
        }
        void deposit(int n){    balance+=n;     }
        void interest(){    balance+=(balance*0.20);    }
        void show(){
            printf("Current balance:%.2f\n",balance);
```

```
        }
};
int main(){
    cout<<"Enter type of account(s for savings, c for current):"<<endl;
    char c;     cin>>c;
    cout<<"Enter balance:"<<endl;
    int bal;     cin>>bal;
    if(c=='c'){
                cout<<"Enter minimum balance:"<<endl;
                int minbal;      cin>>minbal;
                Cur_acct c1;
                c1.setbal(bal,minbal);
                cout<<"Enter amount to deposit:"<<endl;
                int n;  cin>>n;
                cout<<"Do you want to use cheque or cash?(c for cheque)..
                "<<endl;
                char c; cin>>c;
                if(c=='c')c1.cheque();
                c1.deposit(n);  cout<<"Deposited: Rs."<<n<<"\n";
                c1.show();
                cout<<"Enter amount to withdraw:"<<endl;
                cin>>n; c1.withdrawal(n);
                cout<<"Withdrawal: Rs."<<n<<"\n";        c1.show();
                cout<<"Enter amount to withdraw:"<<endl;
                cin>>n; c1.withdrawal(n);
                cout<<"Withdrawal: Rs."<<n<<"\n";        c1.show();
         }
    else{
                Sav_acct s1;
                s1.setbal(bal);
                cout<<"Enter amount to deposit:"<<endl;
                int n;  cin>>n;
                s1.deposit(n);
                cout<<"Deposited: Rs."<<n<<"\n";         s1.show();
                cout<<"Enter amount to withdraw:"<<endl;
        cin>>n; s1.withdrawal(n);
                cout<<"Withdrawal: Rs."<<n<<"\n";
                s1.show();
        cout<<"Enter amount to withdraw:"<<endl;
        cin>>n; s1.withdrawal(n);
                cout<<"Withdrawal: Rs."<<n<<"\n";        s1.show();
                cout<<"Adding interest of 0.20 to current balance:"
                <<endl;
                s1.interest();  s1.show();
    }
    return 0;
}
```

Output:





.

16. Rewrite the above program using constructors.

Program:

```
#include<iostream>
#include<cstdio>
#include<string>
#include<stdlib.h>
#include<cmath>
#define ll long long int
using namespace std;
using namespace std;
class Account{
    string customer_name,acct_type;
    ll acct;
    public:
    Account(){}
    Account(string a,string b,ll c){
        customer_name=a;
        acct_type=b;
        acct=c;
    }
};
class Cur_acct:public Account{
    double balance,minBal;
    public:
        Cur_acct(double a,double b){
            balance=a;
            minBal=b;
        }
        void withdrawal(int n){
            if(balance<minBal)balance-=(balance*0.10);
            else if(n>balance)printf("Insufficient balance.\n");
            else balance-=n;
        }
```

```
        void deposit(int n){    balance+=n;     }
        void cheque(){
            cout<<"Deposited by cheque..."<<endl;
        }
        void show(){
            printf("Current balance=%.2f\n",balance);
        }
};
class Sav_acct:public Account
{
    double balance;
    public:
        Sav_acct(int a){    balance=a;  }
        void withdrawal(int n){
            if(n>balance)printf("Insufficient balance.\n");
            else balance-=n;
        }
        void deposit(int n){    balance+=n;     }
        void interest(){    balance+=(balance*0.20);    }
        void show(){
            printf("Current balance:%.2f\n",balance);
        }
};
int main(){
    cout<<"Enter type of account(s for savings, c for current):"<<endl;
    char c;    cin>>c;
    cout<<"Enter balance:"<<endl;
    int bal;    cin>>bal;
    if(c=='c'){
        cout<<"Enter minimum balance:"<<endl;
        int minbal;     cin>>minbal;
        Cur_acct c1(bal,minbal);
        cout<<"Enter amount to deposit:"<<endl;
        int n;  cin>>n;
        cout<<"Do you want to use cheque or cash?(c for cheque)..."<<endl;
        char c; cin>>c;
        if(c=='c')c1.cheque();
        c1.deposit(n);
        cout<<"Deposited: Rs."<<n<<"\n";        c1.show();
        cout<<"Enter amount to withdraw:"<<endl;
        cin>>n; c1.withdrawal(n);
        cout<<"Withdrawal: Rs."<<n<<"\n";
        c1.show();
        cout<<"Enter amount to withdraw:"<<endl;        cin>>n;
        c1.withdrawal(n);
        cout<<"Withdrawal: Rs."<<n<<"\n";       c1.show();
    }
    else{
        Sav_acct s1(bal);
        cout<<"Enter amount to deposit:"<<endl;
        int n;  cin>>n;
        s1.deposit(n);
```

```
        cout<<"Deposited: Rs."<<n<<"\n";          s1.show();
        cout<<"Enter amount to withdraw:"<<endl;
    cin>>n;       s1.withdrawal(n);
        cout<<"Withdrawal: Rs."<<n<<"\n";
        s1.show();
        cout<<"Enter amount to withdraw:"<<endl;          cin>>n;
        s1.withdrawal(n);
        cout<<"Withdrawal: Rs."<<n<<"\n";          s1.show();
        cout<<"Adding interest of 0.20 to current balance:"<<endl;
        s1.interest();  s1.show();
    }
    return 0;
}
```

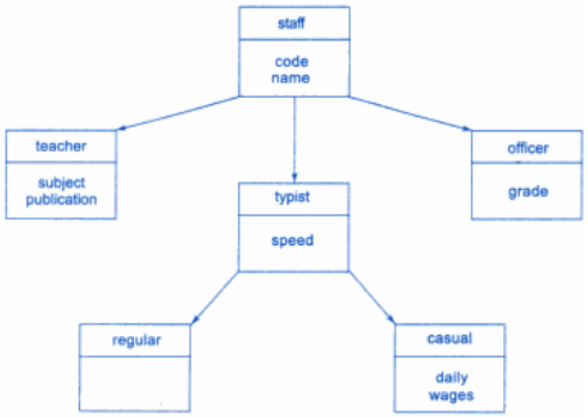
Output:

```
mouri@intel10-hub: ~/c++-files/1
mouri@intel10-hub ~/c++-files/11-23 ./a.out
Enter type of account(s for savings, c for current):
s
Enter balance:
12000
Enter amount to deposit:
2000
Deposited: Rs.2000
Current balance:14000.00
Enter amount to withdraw:
20000
Insufficient balance.
Withdrawal: Rs.20000
Current balance:14000.00
Enter amount to withdraw:
5000
Withdrawal: Rs.5000
Current balance:9000.00
Adding interest of 0.20 to current balance:
Current balance:10800.00
```

```
mouri@intel10-hub: ~/c++-files/1
mouri@intel10-hub ~/c++-files/11-23 ./a.out
Enter type of account(s for savings, c for current):
c
Enter balance:
12000
Enter minimum balance:
2000
Enter amount to deposit:
2000
Do you want to use cheque or cash?(c for cheque)..
c
Deposited by cheque...
Deposited: Rs.2000
Current balance=14000.00
Enter amount to withdraw:
13000
Withdrawal: Rs.13000
Current balance=1000.00
Enter amount to withdraw:
2000
Withdrawal: Rs.2000
Current balance=900.00
```

.

17. An educational institution wishes to maintain a database of its employees. The database is divided into a number of classes whose hierarchical relationships are shown in figure below. The figure also shows the minimum information required for each class. Specify all the classes and define functions to create the database and retrieve individual information as and when required.



Program:

```
#include<iostream>
#include<cstdio>
#include<string>
#include<stdlib.h>
#define ll long long int
using namespace std;
class Staff{
    string code,name;
    public:
    Staff(){}
    void init(string a,string b){  code=a;  name=b;  }
    void getS(){
        cout<<"Code= "<<code<<endl;
        cout<<"Name= "<<name<<endl;
    }
};
class Teacher:public Staff{
    string subj,publ;
    public:
    Teacher(){}
    void init1(string c,string d,string a,string b){
                init(c,d);  subj=a;  publ=b;
    }
    void getT(){
        getS(); cout<<"Subject= "<<subj<<endl;
        cout<<"Publication= "<<publ<<endl;
    }
};
class Typist:public Staff{
    double speed;
    public:
    Typist(){}
    void init1(string a,string b,double d){
                init(a,b);      speed=d;
    }
    void getT2(){
        getS();
        cout<<"Speed= "<<speed<<" Words per minute."<<endl;
    }
};
class Officer:public Staff{
    string grade;
    public:
    Officer(){}
    void init1(string b,string c,string a){
                init(b,c);      grade=a;
    }
    void getO(){
                getS(); cout<<"Grade= "<<grade<<"\n";
    }
```

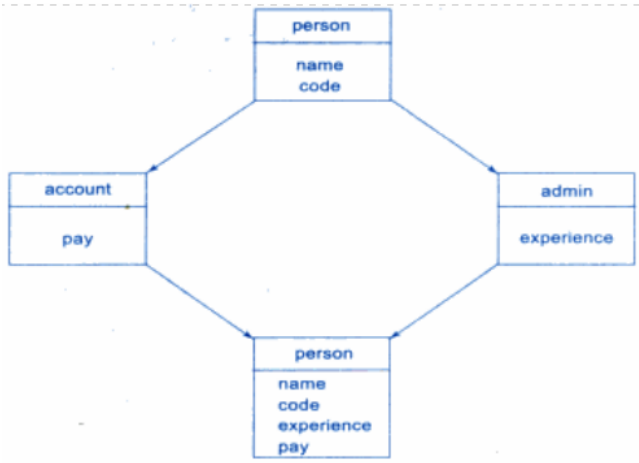
```
};
class Regular:public Typist{
    public:
    Regular(){}
    void init2(string a,string b,int s){  init1(a,b,s);  }
    void getR(){        getT2();    }
};
class Casual:public Typist
{
    int wages;
    public:
    Casual(){}
    void init2(string a,string b,int s,int t)
    {
        init1(a,b,s);
        wages=t;
    }
    void getC()
    {
        getT2();
        cout<<"Daily Wages= "<<wages<<endl;
    }
};
int main()
{
    Teacher t;
    cout<<"Staff: Teacher"<<endl;
    t.init1("0001","A","Maths","Srijan");
    t.getT();
    Officer o;
    cout<<"Staff: Officer"<<endl;
    o.init1("1001","B","A");
    o.getO();
    cout<<"Staff: Typist(Regular)"<<endl;
    Regular r;
    r.init2("2001","C",30);
    r.getR();
    cout<<"Staff: Typist(Casual)"<<endl;
    Casual c;
    c.init2("2002","D",30,10000);
    c.getC();
    return 0;
}
```

Output:

```
mouri@intell0-hub: ~/c++-files/11-23 83x45
mouri@intell0-hub ~/c++-files/11-23 ./a.out
Staff: Teacher
Code= 0001
Name= A
Subject= Maths
Publication= Srijan
Staff: Officer
Code= 1001
Name= B
Grade= A
Staff: Typist(Regular)
Code= 2001
Name= C
Speed= 30 Words per minute.
Staff: Typist(Casual)
Code= 2002
Name= D
Speed= 30 Words per minute.
Daily Wages= 10000
```

.

18. Consider a class network as shown below. Define all four classes and write a program to create, update and display the information contained in master objects.



Program:

```
#include<iostream>
#include<cstdio>
#include<string>
#include<stdlib.h>
#include<cmath>
#define ll long long int
using namespace std;
class Person{
    public:
    string name,code;
    void init(string a,string b){
        name=a; code=b;
    }
    virtual void display()=0;
};
class Account:virtual public Person{
    public:
    int pay;
```

```
    void init1(string a,string b,int c){
        init(a,b);      pay=c;
    }
    void display(){
        cout<<"Name= "<<name<<endl;      cout<<"Code= "<<code<<endl;
        cout<<"Pay= "<<pay<<endl;
    }
};
class Admin:virtual public Person{
    public:
    string exp;
    void init2(string a,string b,string c){
        init(a,b);      exp=c;
    }
    void display(){
        cout<<"Name= "<<name<<endl;      cout<<"Code= "<<code<<endl;
        cout<<"Experience= "<<exp<<endl;
    }
};
class Master:public Account,public Admin{
    public:
    void init3(string a,string b,string c,int d){
        init1(a,b,d);   init2(a,b,c);
    }
    void display(){
        cout<<"Name= "<<name<<endl;      cout<<"Code= "<<code<<endl;
        cout<<"Experience= "<<exp<<endl;  cout<<"Pay= "<<pay<<endl;
    }
};
int main(){
    Master p;
    p.init3("A","0001","2 years",10000);     p.display();
    return 0;
}
```

Output:

```
mouri@intell0-hub ~/c++-files/11-23 ./a.out
Name= A
Code= 0001
Experience= 2 years
Pay= 10000
```

.

19. Write a program demonstrating overriding using 'virtual'.

Program:

```
#include<iostream>
#include<cstdio>
#include<string>
#include<stdlib.h>
#include<cmath>
#define ll long long int
using namespace std;
```

```
class Base{
    public:
        void display(){     cout<<"Display Base"<<endl;   }
        virtual void show(){     cout<<"Show Base"<<endl;          }
};
class Derived:public Base{
    public:
        void display(){     cout<<"Display Derived"<<endl;        }
        void show(){     cout<<"Show Derived"<<endl;      }
};
int main(){
    Base b;
    Derived d;     Base *bptr;
    bptr=&b;
    bptr->display();     bptr->show();
    bptr=&d;
    bptr->display();     bptr->show();
    return 0;
}
```

Output:

```
mouri@intell0-hub ~/c++-files/11-30 ./a.out
Display Base
Show Base
Display Base
Show Derived
```

.

20. Write a program implementing Shape class, from which Triangle and Rectangle inherit. Use 'virtual'

Program:

```
#include<iostream>
#include<cstdio>
#include<string>
#include<stdlib.h>
#include<cmath>
#define ll long long int
using namespace std;
class Shape{
    public:
    double l,b;
    void getData(double _l,double _b){
        l=_l;    b=_b;
    }
    virtual void display(){
        cout<<"Area ="<<(l*b)<<"\n";
    }
};
class Triangle:public Shape{
    public:
    void display(){
```

```
        cout<<"Area of triangle="<<(0.5*b*l)<<"\n";
    }
};
class Rectangle:public Shape{
    public:
    void display(){
                cout<<"Area of triangle="<<(l*b)<<"\n";
    }
};
class Circle:public Shape{
    public:
    void getData(int _l,int _b=0){
        l=_l;   b=_b;
    }
    void display(){
        cout<<"Area of circle="<<(3.1412*l*l)<<"\n";
        }
};
int main()
{
    Triangle t; Rectangle r;
    Circle c;
    double l,b;
    t.getData(4,5);      t.display();
    r.getData(4,5);      r.display();
    c.getData(5);        c.display();
    return 0;
}
```

Output:

```
mouri@intel10-hub ~/c++-files/11-30 ./a.out
Area of triangle=10
Area of rectangle=20
Area of circle=78.53
```

.

21. Rewrite the previous program without using 'virtual'.

Program:

```
#include<iostream>
#include<cstdio>
#include<string>
#include<stdlib.h>
#include<cmath>
#define ll long long int
using namespace std;
class Shape{
    public:
    double l,b;
    void getData(double _l,double _b){
        l=_l;   b=_b;
    }
```

```
    void display(){
        cout<<"Area ="<<(l*b)<<"\n";
    }
};
class Triangle:public Shape{
    public:
    void display(){
        cout<<"Area of triangle="<<(0.5*b*l)<<"\n";
    }
};
class Rectangle:public Shape{
    public:
    void display(){
        cout<<"Area of rectangle="<<(l*b)<<"\n";
    }
};
class Circle:public Shape{
    public:
    void getData(int _l,int _b=0){
        l=_l;   b=_b;
    }
};
int main()
{
    Triangle t; Rectangle r;
    Circle c;
    t.getData(4,5);      t.display();
    r.getData(4,5);      r.display();
    c.getData(5);        c.display();
    return 0;
}
```

Output:

```
mouri@intel0-hub ~/c++-files/11-30 ./a.out
Area of triangle=10
Area of rectangle=20
Area =0
```

.

22. Write a function template for finding the minimum value contained in an array.

Program:

```
#include<iostream>
#include<cstdio>
#include<string>
#include<stdlib.h>
#include<cmath>
#define ll long long int
using namespace std;
template<class R>
R minimum(R a[],int n){
```

```
    R m;      m=a[0];
    for(int i=1;i<3;i++)if(m>a[i])m=a[i];
    return m;
}
int main(){
    int x[3]={10,21,3};
    cout<<minimum(x,3)<<"\n";
    return 0;
}
```

Output:

```
mouri@intell0-hub ~/c++-files/12-7 ./a.out
3
```

.

23. Write a program containing a possible exception. Perform exception handling with multiple catch.

Program:

```
#include<iostream>
#include<cstdio>
#include<string>
#include<stdlib.h>
#include<cmath>
#define ll long long int
using namespace std;
void excp(){
    int x=4;     int y=4;
    if(x==y)throw(x-y);
    else cout<<"Its fine"<<endl;
}
int main(){
    try{         excp();     }
    catch(int i){
        cout<<"Both are equal."<<endl;
    }
    catch(char c){
        cout<<"Character is found."<<endl;
    }
    return 0;
}
```

Output:

```
mouri@intell0-hub ~/c++-files/12-7 ./a.out
Both are equal.
```

.

24. Write a program to demonstrate the concept of rethrowing an exception.

Program:

```
#include<iostream>
#include<cstdio>
#include<string>
#include<stdlib.h>
#include<cmath>
#define ll long long int
using namespace std;
void excp1(){
    int x=9;    int y=7;
    if(x-y==2)throw('A');
    else cout<<"Its fine."<<endl;
}
int main(){
    try{        excp1();    }
    catch(char c){
        if(c=='B')cout<<"The diff is 2."<<endl;
        else throw;
    }
    return 0;
}
```

Output:

```
mouri@intel10-hub ~/c++-files/12-7 ./a.out
terminate called after throwing an instance of 'char'
[5]    12111 abort (core dumped)  ./a.out
```

.

25. Write a class template to represent a generic vector. Include member functions to perform the following taks:
* to create a vector
* to modify the value of a given element
* to multiply by a scalar.

Program:

```
#include<iostream>
#include<cstdio>
#include<string>
#include<stdlib.h>
#include<cmath>
#define ll long long int
using namespace std;
template<class T>
class vector
{
    T *v;
    public:
        void init(int size){
```

```
            v=new int[size];
            cout<<"Enter values"<<endl;
            for(int i=0;i<size;i++)cin>>v[i];
        }
        T multiply(vector &a,vector &b){
            T sum=0;
            for(int i=0;i<3;i++)sum+=a.v[i]*b.v[i];
            return sum;
        }
};
int main(){
    vector<int> v1;      vector<int> v2;
    vector<int> v;
    cout<<"Enter 3 values for v1:"<<endl;       v1.init(3);
    cout<<"Enter 3 values for v2:"<<endl;       v2.init(3);
    cout<<"Product="<<v.multiply(v1,v2)<<"\n";
    return 0;
}
```

Output:

```
mouri@intell0-hub ~/c++-files/12-7 ./a.out
Enter 3 values for v1:
Enter values
1 2 3
Enter 3 values for v2:
Enter values
4 5 6
Product=32
```

.