Q1.

#include<iostream>

using namespace std;

class Apple

{

protected:

int a;

public:

void getA()

{

cout<<"\nEnter value of Apple: ";

cin>>a;

}

};

class Mango

{

protected:

int b;

public:

void getB()

{

cout<<"\nEnter value of Mangoes: ";

cin>>b;

}

};

class Fruit:public Apple,public Mango

{

public:

void display()

{

cout<<"Number of Apples in fruit basket: "<<a<<endl;

cout<<"Number of Mangoes in fruit basket: "<<b<<endl;

cout<<"Total sum of fruits : "<<a+b;

}

};

int main()

{

Fruit c;

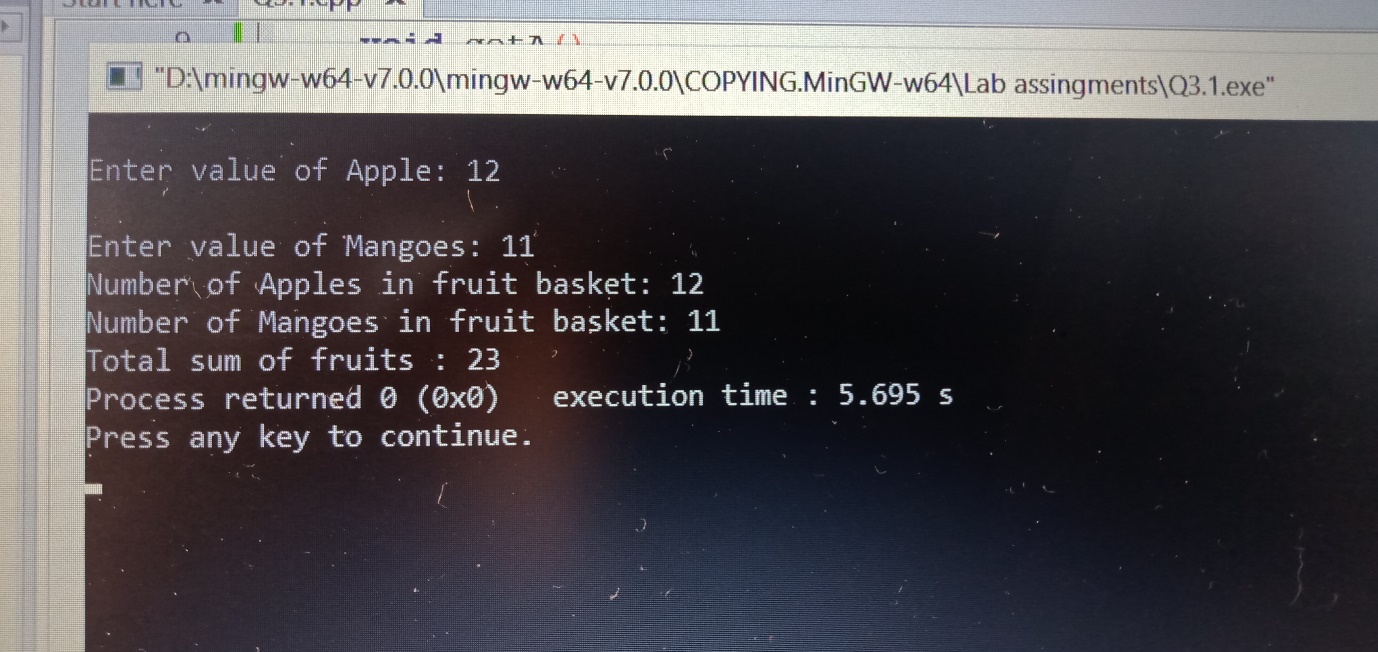
c.getA();

c.getB();

c.display();

return 0;

}



Q2

#include<iostream>

using namespace std;

class marks

{

public:

int rno,m1,m2,m3;

char nm[25];

};

class physics:public marks

{

public:

void get()

{

cout<<"\nPhysics mark: ";

cin>>m1;

}

int display()

{

return m1;

}

};

class Mathematics:public marks

{

public:

void get()

{

cout<<"\nMathematics mark: ";

cin>>m2;

}

int display()

{

return m2;

}

};

class chemistry:public marks

{

public:

void get()

{

cout<<"\nChemistry mark: ";

cin>>m3;

}

int display()

{

return m3;

}

};

int main()

{

marks m;

physics p;

Mathematics mt;

chemistry c;

int n;

cout<<"\nEnter of student class: ";

cin>>n;

int tot=0,i;

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

{

cout<<"\nRoll no: "<<i+1;

p.get();

mt.get();

c.get();

tot+=p.display()+mt.display()+c.display();

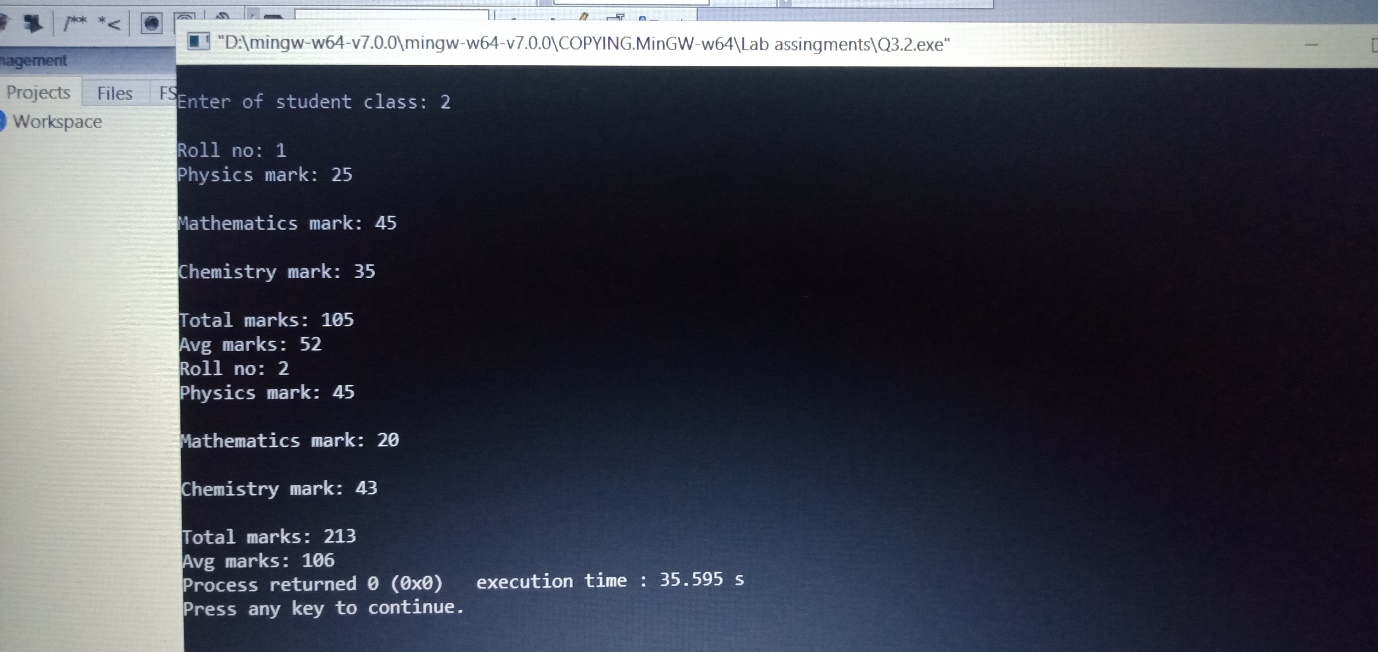
cout<<"\nTotal marks: "<<tot;

cout<<"\nAvg marks: "<<tot/n;

}

return 0;

}



Q3

#include<iostream>

using namespace std;

#include<math.h>

class Account

{

public:

long int acno;

string Cnm,AcType;

int deposite,balance,withdraw;

void setdata()

{

cout<<"\nEnter account no: ";

cin>>acno;

cout<<"\nEnter name: ";

cin>>Cnm;

cout<<"\nEnter Account type: ";

cin>>AcType;

}

void Deposite()

{

cout<<"\nEnter Amount for Deposite: ";

cin>>deposite;

balance=deposite;

}

void Withdraw()

{

cout<<"\nEnter Amount for Withdraw: ";

cin>>withdraw;

if(balance<=withdraw)

{

cout<<"\nSorry your balance is not sufficient for withdraw ";

}

else

{

balance=balance-withdraw;

cout<<"\nYour withdraw is successful. ";

}

cout<<"\nYour current balance: "<<balance;

}

void display()

{

cout <<"\n Account Holder Name: "<<Cnm<<endl;

cout <<"\nAccount no: "<<acno<<endl;

cout <<"\nAccount type: "<<AcType<<endl;

cout <<"\nYour Balance: "<<balance<<endl;

}

};

class savingAc:public Account

{

public:

float comp\_intrest,dep\_intrest;

void compintrest()

{

float p,rate,t,comp\_intrest;

cout<<"\nEnter principle: ";

cin>>p;

cout<<"\nEnter rate: ";

cin>>rate;

cout<<"\nEnter time in year: ";

cin>>t;

comp\_intrest=p\*pow((1+rate/100),t);

cout<<"\nCompund intrest: "<<comp\_intrest;

}

};

class currentAc:public Account

{

public:

int MinBal=500;

void minbalance()

{

if(balance<MinBal)

{

cout<<"\nYour Balance is low!";

}

}

void chequebook()

{

cout<<"\nAccount holder: "<<Cnm<<endl;

cout<<"\nAccount no: "<<acno<<endl;

cout<<"\nAccount Balance: "<<balance<<endl;

}

};

int main()

{

Account a;

savingAc b;

currentAc c;

a.setdata();

a.Deposite();

a.Withdraw();

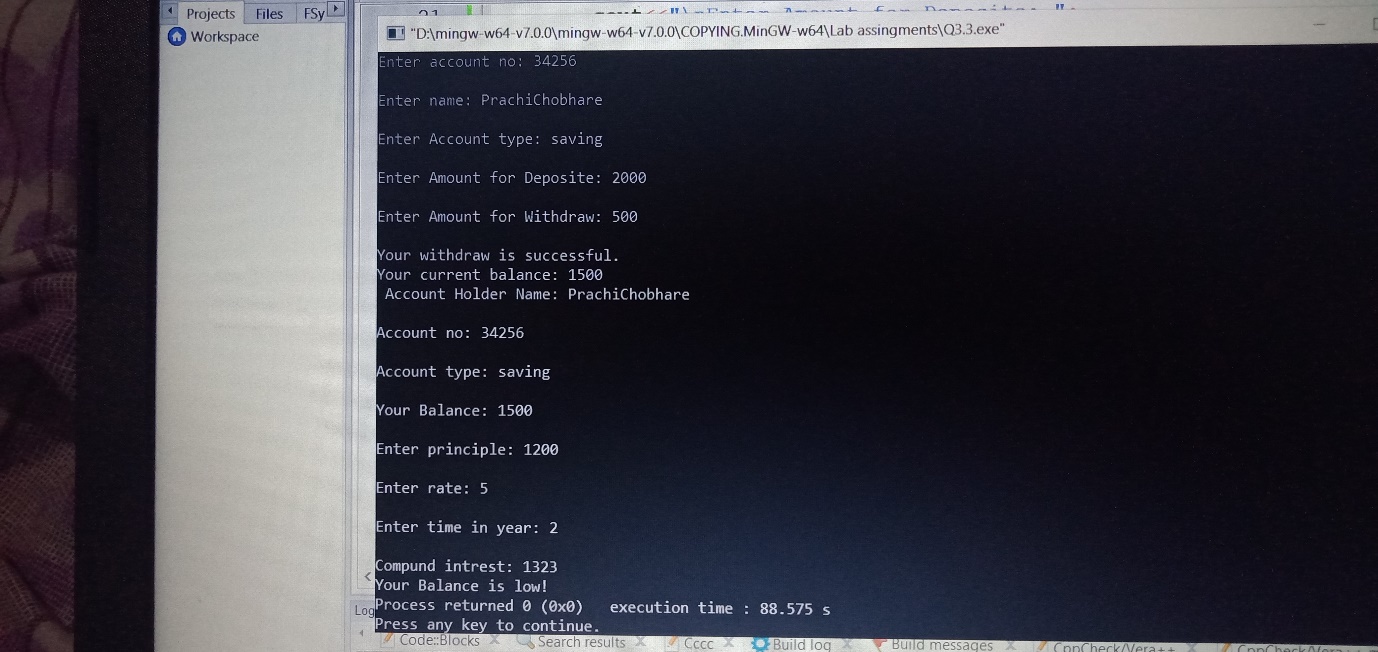
a.display();

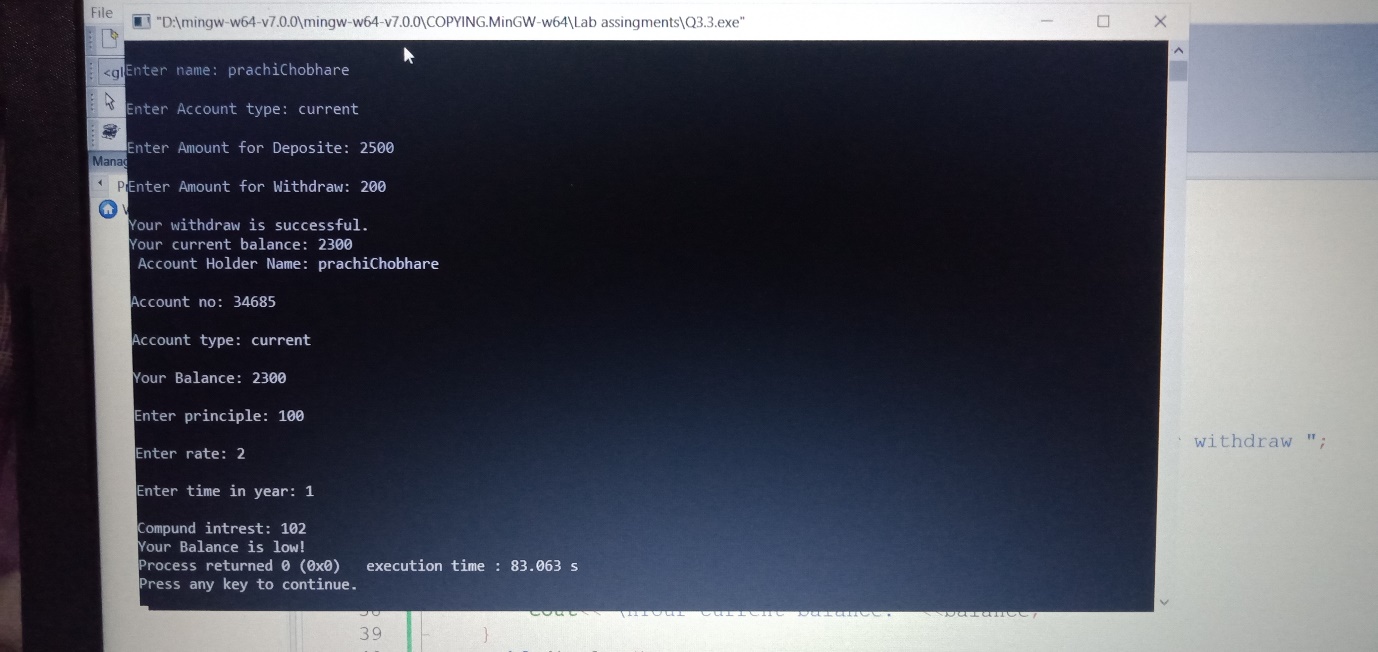
b.compintrest();

c.minbalance();

return 0;

}





Q4

#include<iostream>

using namespace std;

#include<math.h>

class Account

{

public:

long int acno;

string Cnm,AcType;

int deposite,balance,withdraw;

Account()

{

deposite=2000;

withdraw=500;

}

void setdata()

{

cout<<"\nEnter account no: ";

cin>>acno;

cout<<"\nEnter name: ";

cin>>Cnm;

cout<<"\nEnter Account type: ";

cin>>AcType;

}

void Deposite()

{

cout<<"\nEnter Amount for Deposite: ";

cin>>deposite;

balance=deposite;

}

void Withdraw()

{

cout<<"\nEnter Amount for Withdraw: ";

cin>>withdraw;

if(balance<=withdraw)

{

cout<<"\nSorry your balance is not sufficient for withdraw ";

}

else

{

balance=balance-withdraw;

cout<<"\nYour withdraw is successful. ";

}

cout<<"\nYour current balance: "<<balance;

}

void display()

{

cout <<"\n Account Holder Name: "<<Cnm<<endl;

cout <<"\nAccount no: "<<acno<<endl;

cout <<"\nAccount type: "<<AcType<<endl;

cout <<"\nYour Balance: "<<balance<<endl;

}

};

class savingAc:public Account

{

public:

float comp\_intrest,dep\_intrest;

void compintrest()

{

float p,rate,t,comp\_intrest;

cout<<"\nEnter principle: ";

cin>>p;

cout<<"\nEnter rate: ";

cin>>rate;

cout<<"\nEnter time in year: ";

cin>>t;

comp\_intrest=p\*pow((1+rate/100),t);

cout<<"\nCompund intrest: "<<comp\_intrest;

}

};

class currentAc:public Account

{

public:

int MinBal=500;

void minbalance()

{

if(balance<MinBal)

{

cout<<"\nYour Balance is low!";

}

}

void chequebook()

{

cout<<"\nAccount holder: "<<Cnm<<endl;

cout<<"\nAccount no: "<<acno<<endl;

cout<<"\nAccount Balance: "<<balance<<endl;

}

};

int main()

{

Account a;

savingAc b;

currentAc c;

a.setdata();

a.Deposite();

a.Withdraw();

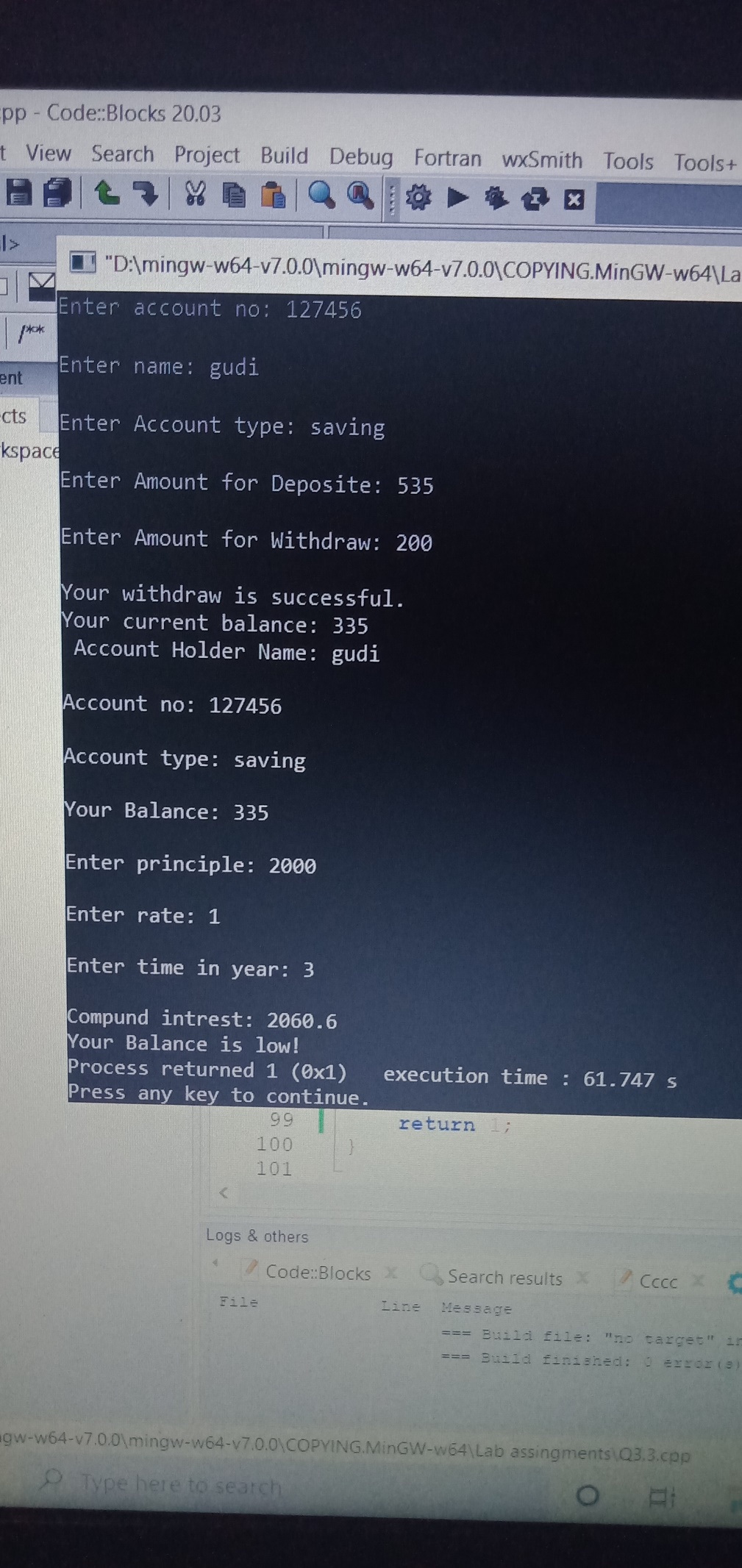
a.display();

b.compintrest();

c.minbalance();

return 1;

}



Q5

#include<iostream>

using namespace std;

class cat

{

protected:

int x;

public:

void get()

{

cout<<"\nEnter num : ";

cin>>x;

}

};

class lion:virtual public cat

{

protected:

int y;

public:

void get1()

{

cout<<"\nEnter num : ";

cin>>y;

}

};

class tiger:virtual public cat

{

protected:

int z;

public:

void get2()

{

cout<<"\nEnter num : ";

cin>>z;

}

};

class Animals:public lion,public tiger

{

public:

void display()

{

get();get1();get2();

cout<<"\nNumber of cats: "<<x;

cout<<"\nNumber of lions: "<<y;

cout<<"\nNumber of tigers: "<<z;

cout<<"\nDivision of animals: "<<x/y/z;

cout<<"\nTotal animals: "<<x+y+z;

cout<<"\nAvg of animals: "<<(x+y+z)/3;

}

};

int main()

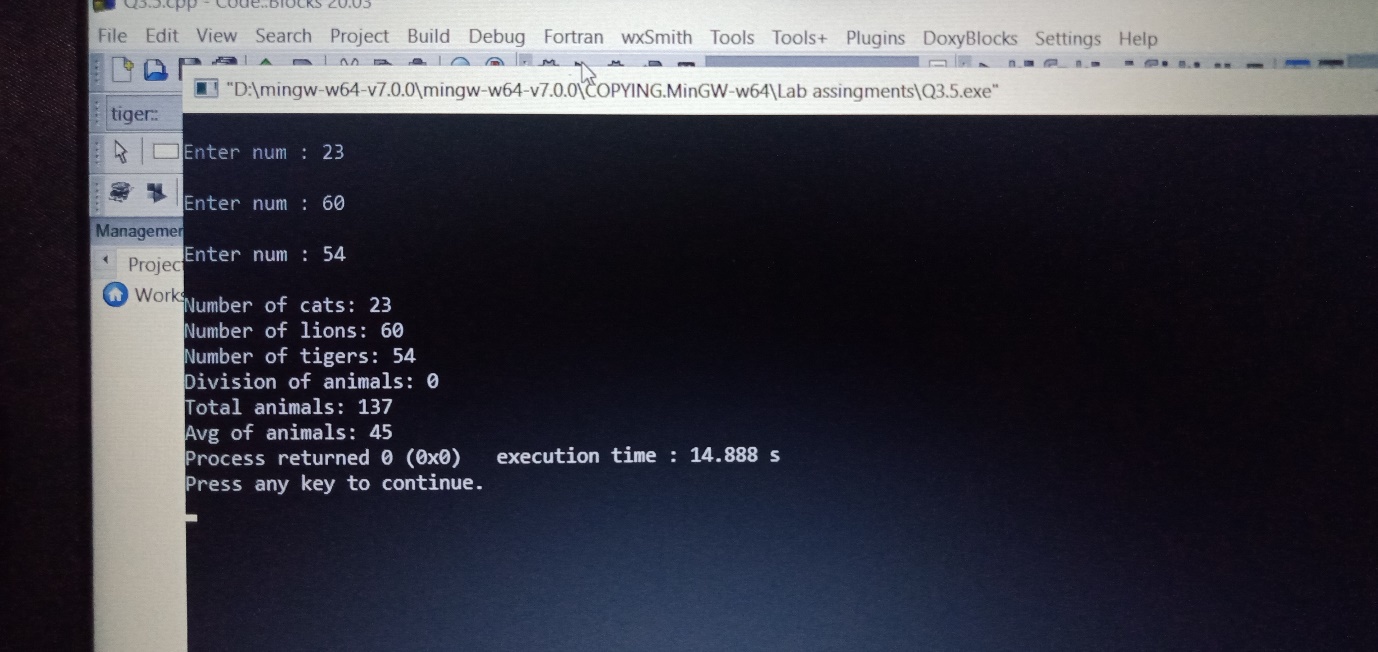
{

Animals a;

a.display();

return 0;

}



Q6

#include<iostream>

using namespace std;

#include<cmath>

class rectangle;

class polar

{

float x,y;

public:

polar()

{

x=0;

y=0;

}

polar(float X,float Y)

{

x=X;

y=Y;

}

void display()

{

cout<<"\nPolar co-ordinates: ";

cout<<"\nX: "<<x<<"\nY: "<<y;

}

float getX()

{

return x;

}

float getY()

{

return y;

}

};

class rectangle

{

float p,q;

public:

rectangle()

{

p=0;

q=0;

}

rectangle(float P,float Q)

{

p=P;

q=Q;

}

void display()

{

cout<<"\nRectangle sides: ";

cout<<"\nP: "<<p<<"\nQ: "<<q<<endl;

}

rectangle(polar d)

{

p=d.getY()\*cos(3.14\*d.getX()/180);

q=d.getY()\*sin(3.14\*d.getX()/180);

}

friend rectangle operator+(rectangle&r1,rectangle&r2)

{

rectangle r3;

r3.p=r1.p+r2.p;

r3.q=r1.q+r2.q;

return r3;

}

operator polar()

{

return polar(atan(q/p),sqrt(p\*p+q\*q));

}

};

int main()

{

float a1,l1,a2,l2;

cout<<"\nX ,Y : ";

cout<<"\nX : ";

cin>>a1;

cout<<"\nY : ";

cin>>l1;

polar p1(a1,l1);

cout<<"\nEnter P,Q : ";

cout<<"\nP: ";

cin>>a2;

cout<<"\nQ: ";

cin>>l2;

polar p2(a2,l2);

polar p3;

rectangle r1,r2,r3;

r1=p1;

r2=p2;

r3=r1+r2;

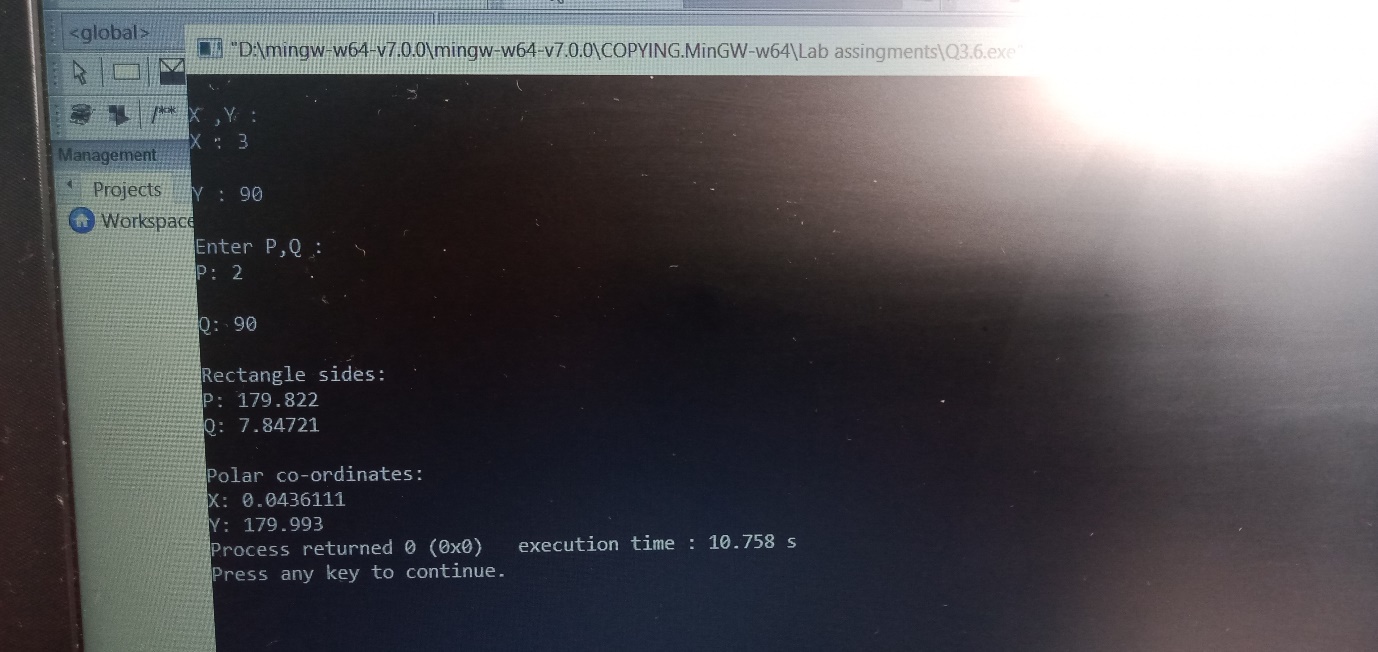
p3=r3;

r3.display();

p3.display();

return 0;

}



#include<iostream>

using namespace std;

class Number

{

private:

double mrk;

public:

Number(float num){mrk=num;}

friend Number operator+(const Number &num,double val);

friend Number operator-(const Number &num,double val);

friend Number operator/(const Number &num,double val);

friend Number operator\*(const Number &num,double val);

double getmrk()const{return mrk;}

};

Number operator+(const Number &num,double val)

{

return Number(num.mrk+val);

}

Number operator-(const Number &num,double val)

{

return Number(num.mrk-val);

}

Number operator/(const Number &num,double val)

{

return Number(num.mrk/val);

}

Number operator\*(const Number &num,double val)

{

return Number(num.mrk\*val);

}

int main()

{

Number num1(69859);

Number num2=num1+2;

cout<<"\nTotal marks: "<<num2.getmrk();

Number num3=num1-2;

cout<<"\nTotal marks: "<<num3.getmrk();

Number num4=num1/2;

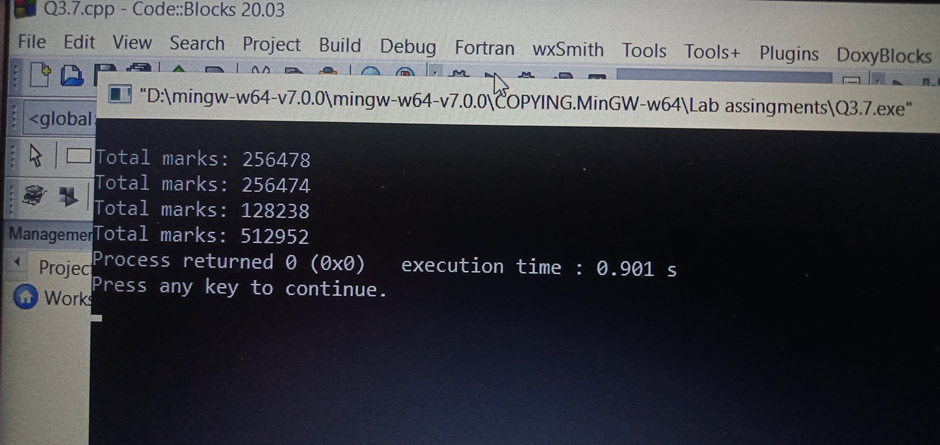
cout<<"\nTotal marks: "<<num4.getmrk();

Number num5=num1\*2;

cout<<"\nTotal marks: "<<num5.getmrk();

return 0;

}



Q8

#include<iostream>

using namespace std;

class MAT

{

int x[3][3],i,j;

public:

void init();

void setdata();

void display();

void add(MAT &,MAT &);

void sub(MAT &,MAT &);

void mul(MAT &,MAT &);

void transpose();

};

void MAT::init()

{

for(int i=0;i<3;i++)

{

for(int j=0;j<3;j++)

{

x[i][j]=0;

}

}

}

void MAT::setdata()

{

for(int i=0;i<3;i++)

{

for(int j=0;j<3;j++)

{

cin>>x[i][j];

}

}

}

void MAT::display()

{

for(int i=0;i<3;i++)

{

for(int j=0;j<3;j++)

{

cout<<x[i][j];

}

cout<<'\t'<<'\n';

}

}

void MAT::transpose()

{

cout<<"\nTranspose of Matrix: ";

for(int i=0;i<3;i++)

{

for(int j=0;j<3;j++)

{

cout<<x[i][j];

}

cout<<'\t'<<'\n';

}

}

void MAT::add(MAT &m,MAT &n)

{

cout<<"\nAddition of matrix: ";

for(int i=0;i<3;i++)

{

for(int j=0;j<3;j++)

{

x[i][j]=(m.x[i][j]+n.x[i][j]);

}

}

}

void MAT::sub(MAT &m,MAT &n)

{

cout<<"\nAddition of matrix: ";

for(int i=0;i<3;i++)

{

for(int j=0;j<3;j++)

{

x[i][j]=(m.x[i][j]+n.x[i][j]);

}

}

}

void MAT::mul(MAT &m,MAT &n)

{

cout<<"\nAddition of matrix: ";

for(int i=0;i<3;i++)

{

for(int j=0;j<3;j++)

{

x[i][j]+=(m.x[i][j]\*n.x[i][j]);

}

}

}

int main()

{

MAT m,n,o;

m.init();

n.init();

m.setdata();

n.setdata();

m.display();

n.display();

m.transpose();

n.transpose();

o.add(m,n);

o.display();

o.sub(m,n);

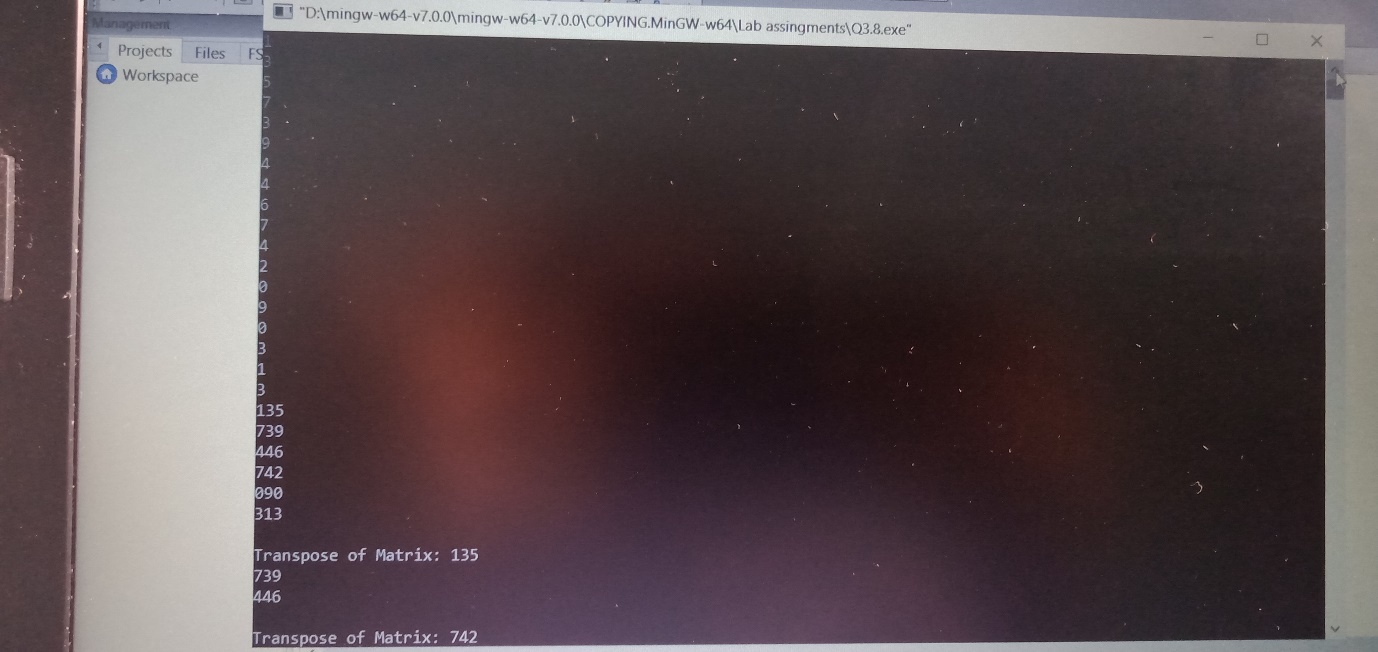
o.display();

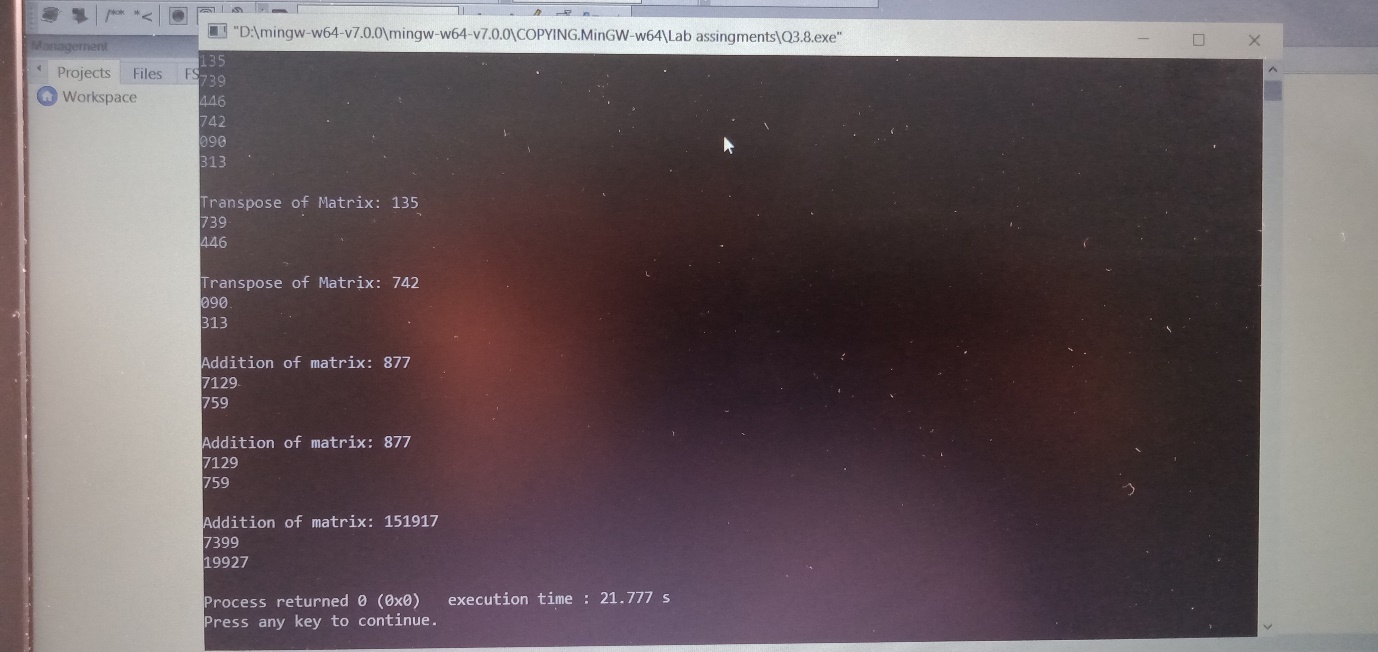
o.mul(m,n);

o.display();

return 0;

}





Q9

#include<string>

#include<cstring>

#include<iostream>

using namespace std;

class Cstring

{

public:

char str[30];

Cstring(char str1[])

{

strcpy(this->str,str1);

}

int operator==(Cstring s2)

{

if(strcmp(str,s2.str)==0)

return 1;

else

return 0;

}

int operator>=(Cstring s3)

{

if(strlen(str)>=strlen(s3.str))

return 1;

else

return 0;

}

};

void compare(Cstring s1,Cstring s2)

{

if(s1==s2)

cout<<s1.str<<" is equal to "<<s2.str<<endl;

else

{

cout<<s1.str<<" is not equal to "<<s2.str<<endl;

}

}

void testcase1()

{

char str1[]="Pooja";

char str2[]="Bandgar";

Cstring s1(str1);

Cstring s2(str2);

compare(s1,s2);

}

void testcase2()

{

char str1[]="PoojaBandgar";

char str2[]="PoojaBanddgar";

Cstring s1(str1);

Cstring s2(str2);

compare(s1,s2);

}

int main()

{

testcase1();

testcase2();

return 0;

}

