Name : Mukul Kumar

Enrollment number : 201B162

Batch : B5

Subject : OOPL

Lab assignment number : 4

Semester : 2

1. The annual examination results of 100 students are tabulated as follows:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Roll No. Subject 1 Subject 2 Subject 3

. . . .

. . . .

. . . .

Write a program to read the data and determine the following:

a) Total marks obtained by each student.

b) The highest marks in each subject and the Roll No. of the students who secured it.

c) The student who obtained the highest total marks.

Code:

#include <iostream>

#include <time.h>

#include <string>

using std::cin;

using std::cout;

using std::endl;

using std::string;

class examination

{

int marks[3];

public:

string roll\_no;

void setmarks(int \*marks);

int total\_marks();

friend void highest\_marks(examination \*);

void print\_all()

{

cout << roll\_no << " " << marks[0] + marks[1] + marks[2] << endl;

}

};

int main()

{

srand(time(0));

examination st[100];

for (int i = 0; i < 100; i++)

{

st[i].roll\_no = "201B" + std::to\_string(i + 1);

int temp[3];

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

{

temp[j] = 7 + rand() % (15 - 7 + 1);

}

st[i].setmarks(temp);

}

for (int i = 0; i < 100; i++)

{

st[i].print\_all();

}

highest\_marks(st);

return 0;

}

void examination::setmarks(int \*marks)

{

this->marks[0] = marks[0];

this->marks[1] = marks[1];

this->marks[2] = marks[2];

}

int examination::total\_marks()

{

int sum = 0;

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

{

sum += marks[i];

}

return sum;

}

void highest\_marks(examination \*st)

{

int high\_s1 = (st + 0)->marks[0],

high\_s1\_index = 0, high\_s2 = (st + 0)->marks[1], high\_s2\_index = 0, high\_s3 = (st + 0)->marks[2], high\_s3\_index = 0, total\_mark = (st + 0)->total\_marks(), high\_total\_index = 0;

for (int i = 0; i < 99; i++)

{

if ((st + i + 1)->marks[0] > high\_s1)

{

high\_s1 = (st + i + 1)->marks[0];

high\_s1\_index = i + 1;

}

if ((st + i + 1)->marks[1] > high\_s2)

{

high\_s2 = (st + i + 1)->marks[1];

high\_s2\_index = i + 1;

}

if ((st + i + 1)->marks[2] > high\_s3)

{

high\_s3 = (st + i + 1)->marks[2];

high\_s3\_index = i + 1;

}

if ((st + i + 1)->total\_marks() > total\_mark)

{

total\_mark = (st + i + 1)->total\_marks();

high\_total\_index = i + 1;

}

}

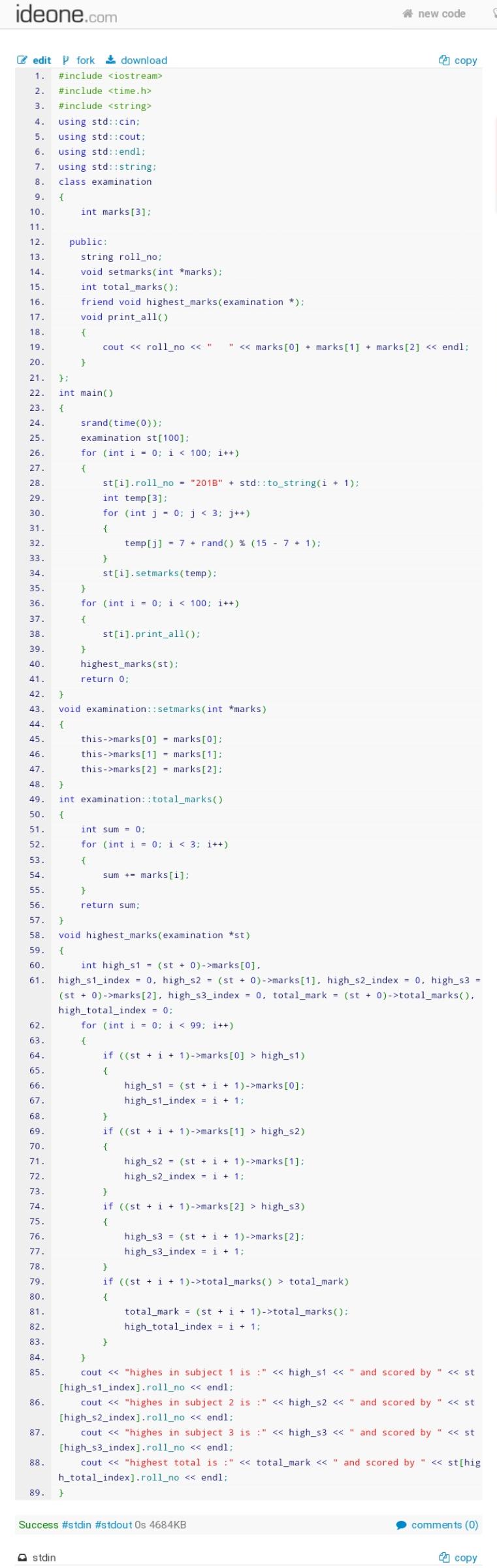
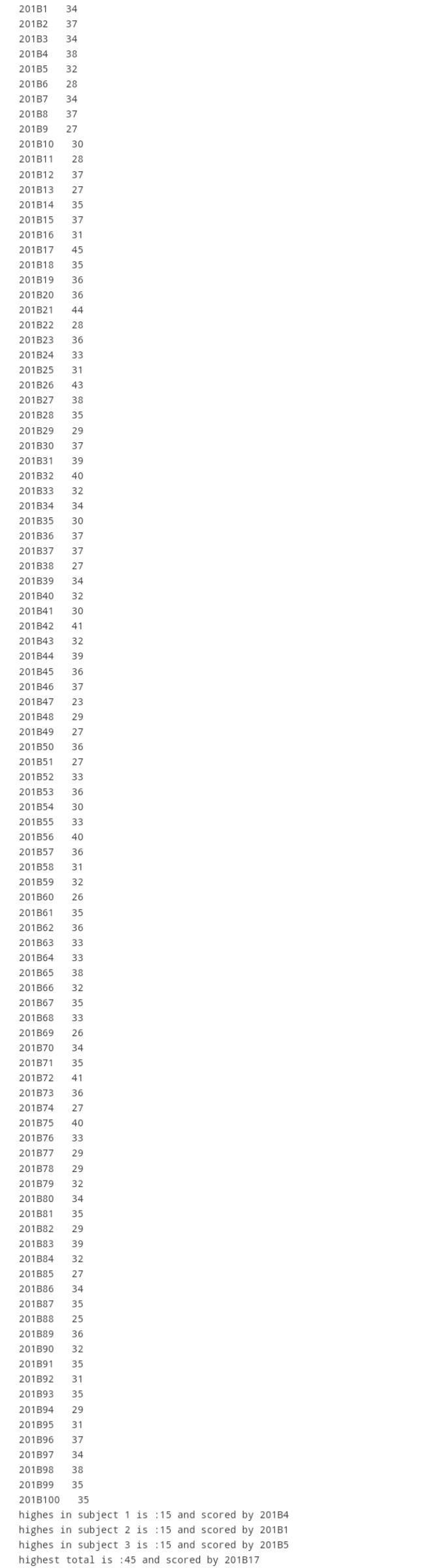
cout << "highes in subject 1 is :" << high\_s1 << " and scored by " << st[high\_s1\_index].roll\_no << endl;

cout << "highes in subject 2 is :" << high\_s2 << " and scored by " << st[high\_s2\_index].roll\_no << endl;

cout << "highes in subject 3 is :" << high\_s3 << " and scored by " << st[high\_s3\_index].roll\_no << endl;

cout << "highest total is :" << total\_mark << " and scored by " << st[high\_total\_index].roll\_no << endl;

}



2. Define a class to represent a bank account, including the following data members:

§ Name of the depositors

§ Account number

§ Type of account

§ Balance amount in the account

and member functions:

§ To assign initial values

§ To deposit an amount

§ To withdraw an amount after checking the balance

§ To display the name and balance

Write a main program to test the program.

Code:

#include <iostream>

#include <time.h>

using std::cin;

using std::cout;

using std::endl;

using std::string;

class Account

{

string name;

long long int account\_number;

string account\_type;

int Balance;

string pin = "0000";

public:

void initial\_account();

void dep\_amount(int);

void withdraw(int);

void bal\_name();

};

int main()

{

srand(time(0));

Account a1;

int n;

a1.initial\_account();

while (1)

{

cout << "Enter your choice :- " << endl

<< "1. Deposit Amount" << endl

<< "2. Withdraw Amount" << endl

<< "3. Check Balance" << endl;

cin >> n;

switch (n)

{

case 1:

cout << "Deposit the amount you want to enter" << endl;

int bal;

cin >> bal;

a1.dep\_amount(bal);

break;

case 2:

{

{

a1.bal\_name();

cout << "Enter amount to withdraw" << endl;

int amount\_withdraw;

cin >> amount\_withdraw;

a1.withdraw(amount\_withdraw);

}

break;

}

case 3:

{

{

a1.bal\_name();

}

break;

}

case -1:

return 0;

default:

{

cout << "wrong input" << endl;

break;

}

}

}

return 0;

}

void Account::initial\_account()

{

string first[10]= {"Mukul","Sonali","Swapnil","Naman","Golden","Anupam","Princy","Anuj","Hello","World"};

string last[5] = {"Rajput","Sinha","Kushwaha","Singh","Yadav"};

name = first[0+(rand()%(10-0+1))]+last[0+(rand()%(5-0+1))];

account\_number = rand();

string ac\_ty[4] = {"Saving", "Student", "Platinum", "Gold"};

account\_type = ac\_ty[0 + (rand() % 3 - 0 + 1)];

Balance = 1000;

}

void Account::dep\_amount(int bal)

{

Balance += bal;

}

void Account::withdraw(int withdrw)

{

bal\_name();

if (Balance - 1000 >= withdrw)

{

cout << "You have withdrawn " << withdrw << endl;

Balance -= withdrw;

cout << "Your new balance is : " << Balance << endl;

}

else

{

cout << "Insufficient Balance" << endl;

}

}

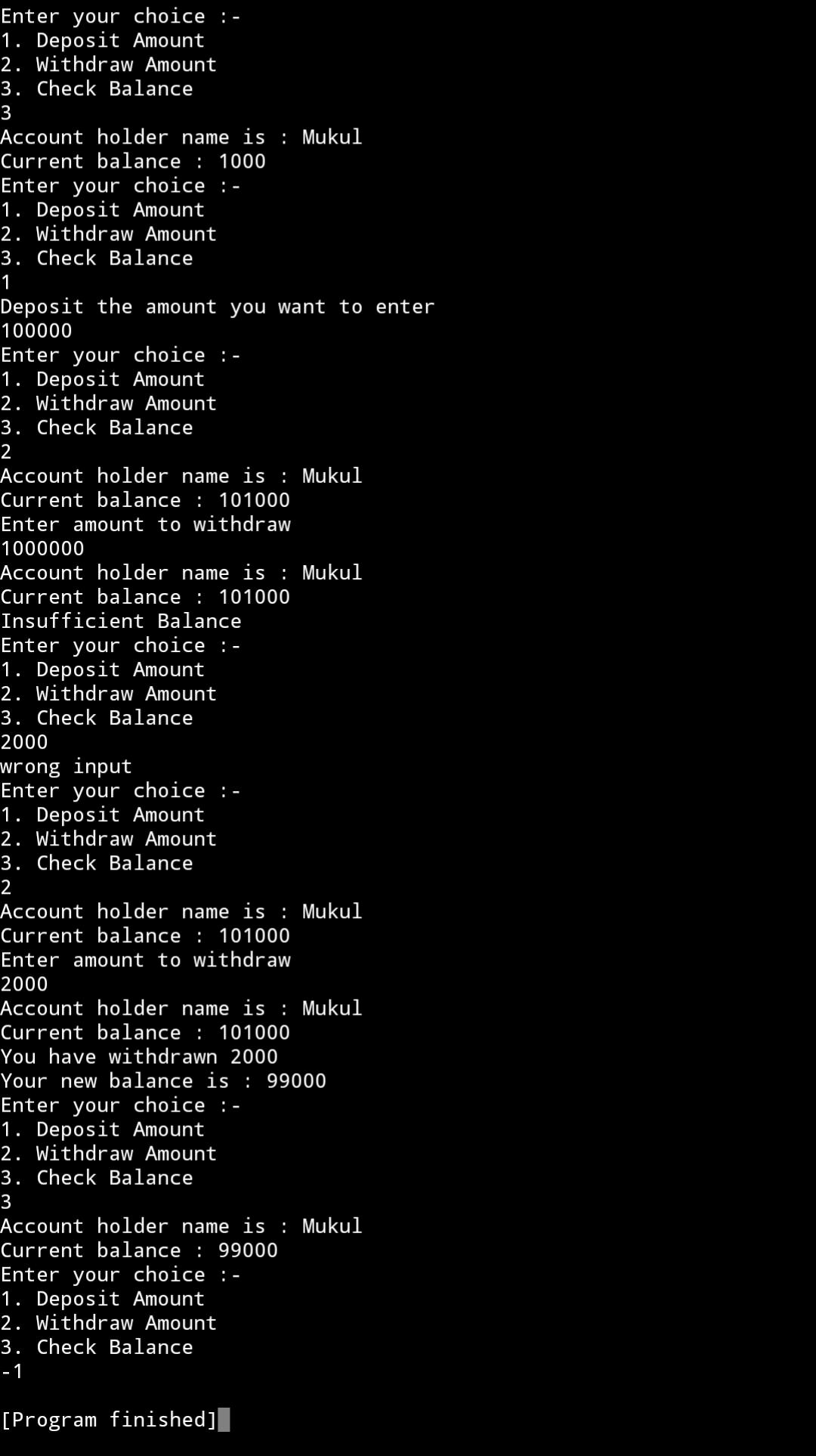
void Account::bal\_name()

{

cout << "Account holder name is : " << name << endl

<< "Current balance : " << Balance << endl;

}



3. Create two classes DM and DB which store the value of the distances. DM stores distances in meters and centimeters and DB in feet and inches. Write a program that can read values for the class objects and add one object of DM with another object of DB.

Code:

#include <iostream>

using std::cin;

using std::cout;

using std::endl;

class DB;

class DM

{

int meter;

int centimeter;

public:

void getInput();

friend void add\_meter(DM, DB);

friend void add\_feet(DM, DB);

};

class DB

{

int feet;

int inch;

public:

void getInput();

friend void add\_meter(DM, DB);

friend void add\_feet(DM, DB);

};

int main()

{

DM a;

DB b;

a.getInput();

b.getInput();

add\_meter(a, b);

add\_feet(a, b);

return 0;

}

void DM::getInput()

{

cout << "Enter meter and centimeter : ";

cin >> meter >> centimeter;

}

void DB::getInput()

{

cout << "Enter feet and inch : ";

cin >> feet >> inch;

}

void add\_meter(DM m, DB f)

{

float x = m.meter + (f.feet / 3.281);

float y = m.centimeter + (x - (int)x) \* 100 + (f.inch \* 2.54);

x += y / 100;

y = ((int)y % 100);

y += (x - (int)x) \* 100;

x = ((int)x);

cout << "In meters " << x << " "

<< "and centimeter " << y << endl;

}

void add\_feet(DM m, DB f)

{

float x = f.feet + (m.meter \* 3.281);

float y = f.inch + (x - (int)x) \* 12 + (m.centimeter / 2.54);

x += (y / 12);

y = (int)y % 12;

y += (x - (int)x) \* 12;

x = (int)x;

cout << "In feets " << x << " "

<< "and inches " << y << endl;

}

