#include<iostream>

using namespace std;

struct Student {

int name;

float roll;

int marks[10];

};

struct Car {

int num;

int\* ptr;

};

struct node {

int data;

node\* next;

};

int main()

{

// Structures are user-defined data types that can contain

// other datatypes as members

Student s1 = {150, 10.2, {10, 30}};

cout << sizeof(s1) << endl;

// Pointers can also be used to access structures

Student \*sptr = &s1;

// The following two statements are equivalent.

// These are two methods of accessing a structure's members

// using a pointer to the structure

cout << (\*sptr).name << endl;

cout << sptr->roll << endl;

int x = 10;

// Structures can also contain pointers as data members

Car c1 = {30, NULL};

// Here the 'ptr' member is being pointed towards x

c1.ptr = &x;

// Assigning a structure pointer to a structure

Car \*cptr = &c1;

// The following two assignments are identical.

// Here the variable x is being updated by first accessing

// the structure using its pointer cptr, and then using

// the ptr member of the structure c1.

cout << x << endl;

\*(cptr->ptr) = 20;

cout << x << endl;

cout << x << endl;

\*((\*cptr).ptr) = 30;

cout << x << endl;

// Structures can also contain pointers to other structures

// of the same type.

node n1 = {10, NULL};

node n2 = {20, NULL};

n1.next = &n2;

// The following two statements are identical.

// Here the first structure is being used to access the

// data element of the second structure through its pointer next

cout << (\*(n1.next)).data << endl;

cout << n1.next->data << endl;

// If we now create a pointer to the first structure n1

node \*head = &n1;

// We can use this pointer to access the data element of the

// first structure n1

// The following two statements are identical

cout << (\*head).data << endl;

cout << head->data << endl;

// We can combine the above two to access the data element of the

// second structure n2 using the pointer to the first structure n1

// The following two statements are identical

cout << (\*(\*head).next).data << endl;

cout << head->next->data << endl;

return 0;

}