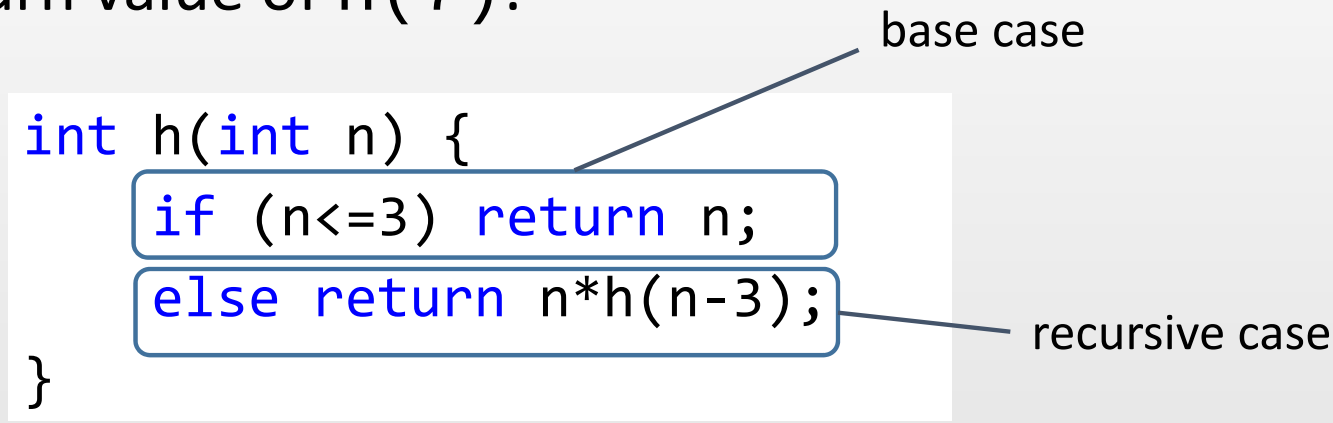
AMA2222 Principles of Programming

Leung Man Kin, Adam
Instructor

adam.leung@polyu.edu.hk
TU720

Quiz 2 review + extra questions

Q1 Suppose h is a recursive function defined as below:
Evaluate the return value of `h(7)`.



The base case suggests:

$$h(1)=1, h(2)=2, h(3)=3$$

The recursive case suggests: $h(7)=7*h(4)$

where $h(4)$ can be evaluated by: $h(4)=4*h(1)=4$

Therefore the return value is: $7*h(4)=7*4=28$

Q2 Suppose `count()` is an integer function with an integer inputs `n` defined as follows. Evaluate the return value of `count(345678)`.

```
int count(int n){
    if (n<10) return 1;
    else return 1+count(n/10);
}
```

base case

recursive case

Since `345678` is >=10, we have to return

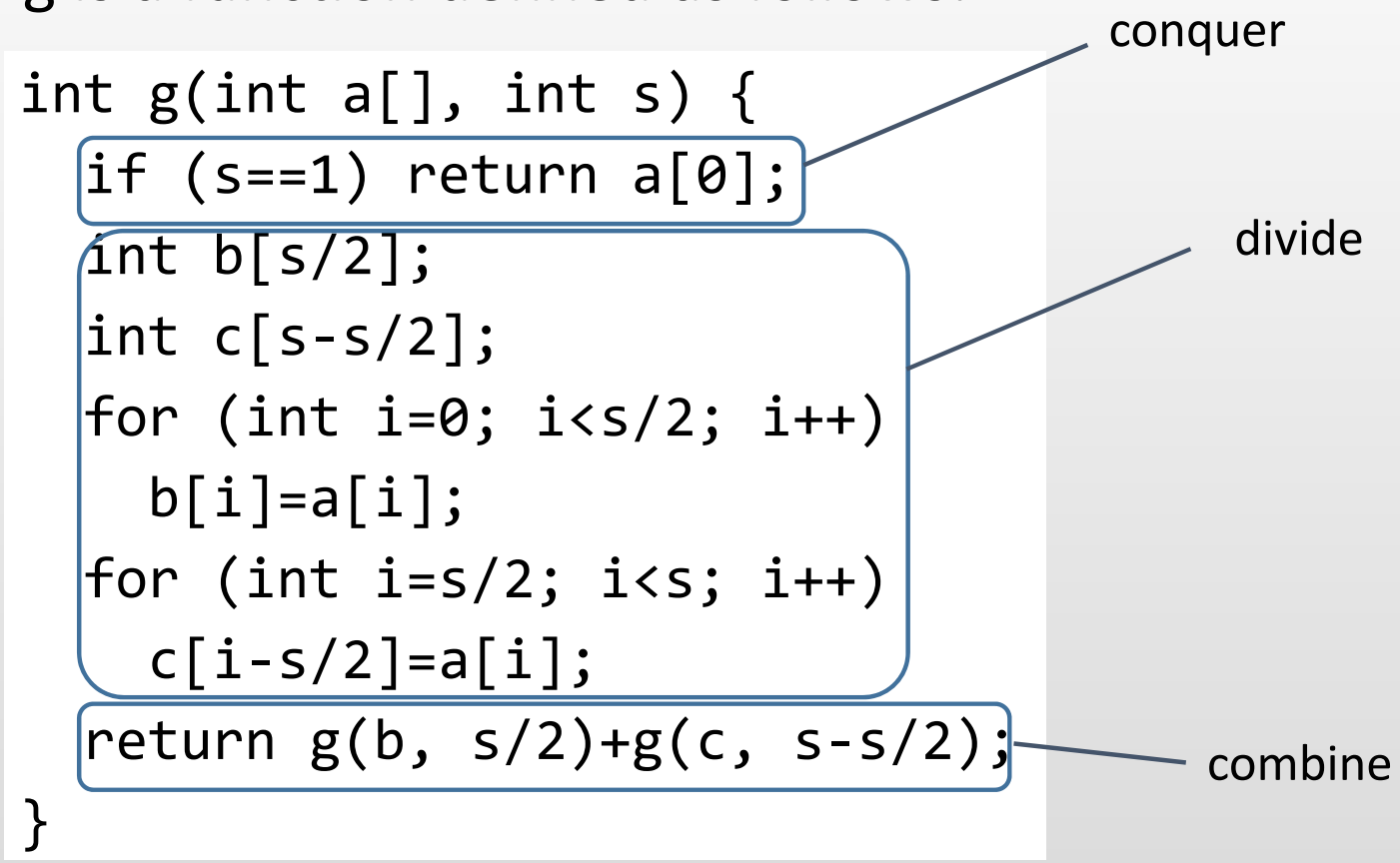
`1+count(34567)`

Since `34567` is >=10, we have to return

`1+(1+count(3456))`

… and so on until we have `count(3)` in the innermost brackets, which simply returns 1. As a result, we have to add up as many 1's as the number of digits. This function actually calculates the number of digits. So the answer is 6.

Q3 Suppose g is a function defined as follows:

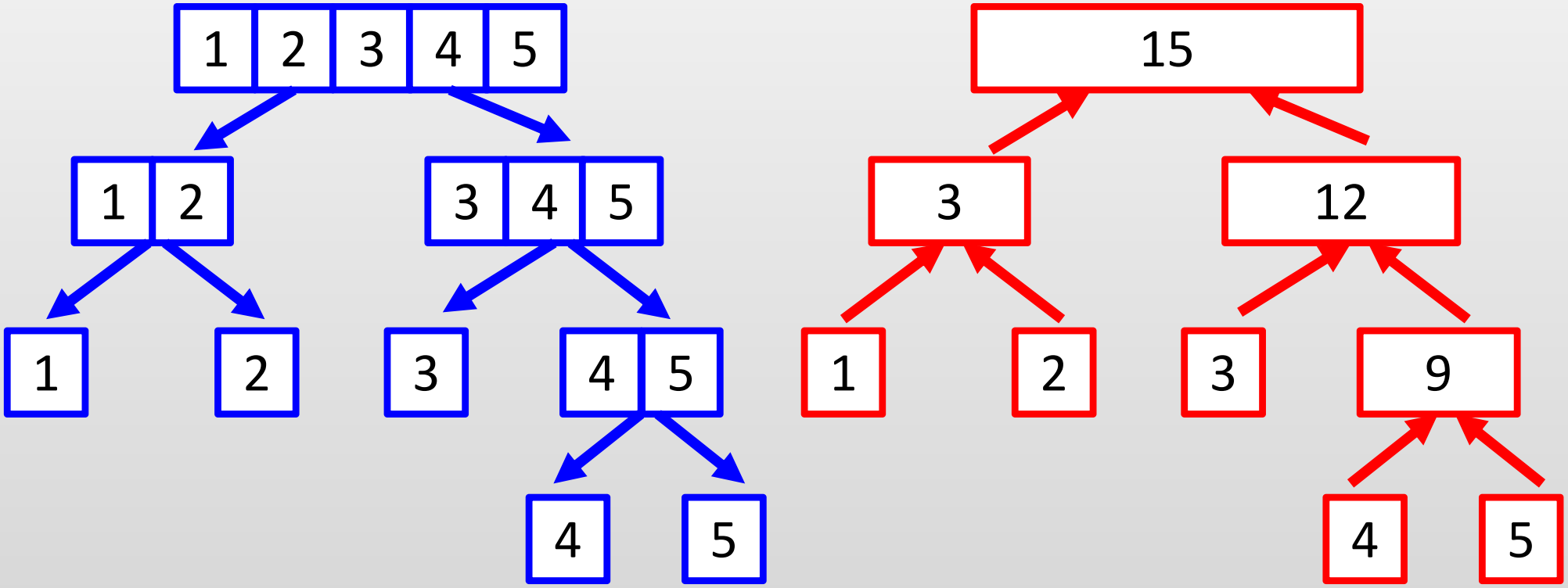


Suppose we have an array:

`int list[] = {1,2,3,4,5}`

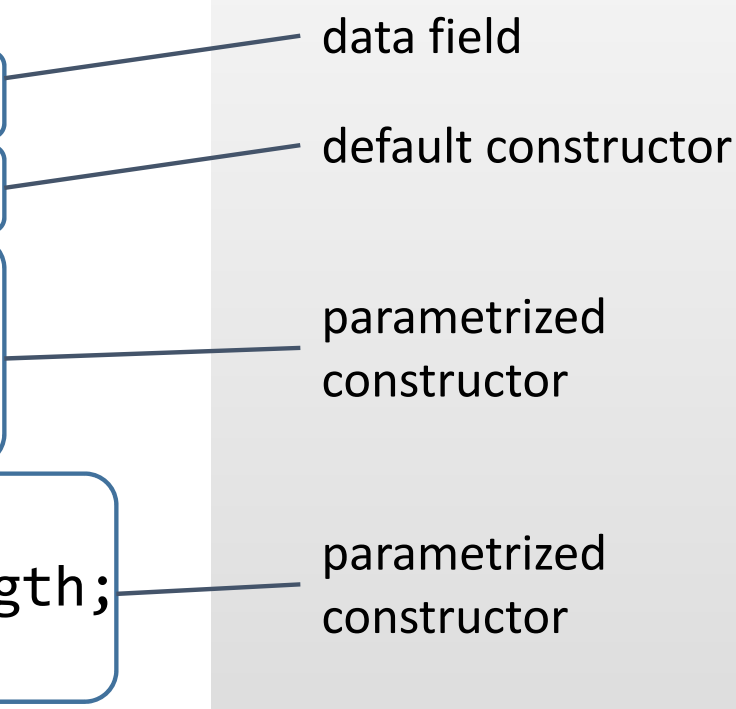
Evaluate the return value of `g(list,5)`.

This function actually add up all the elements in the array, so the result is: 1+2+3+4+5=15



Q4,Q5 A class is defined below:

```
class square{
public:
    double length;
    square() {}
    square(double x){
        length = x;
    }
    double getArea(){
        return length*length;
    }
};
```



A student has defined a square class object as follows:

```
square s(5);
```

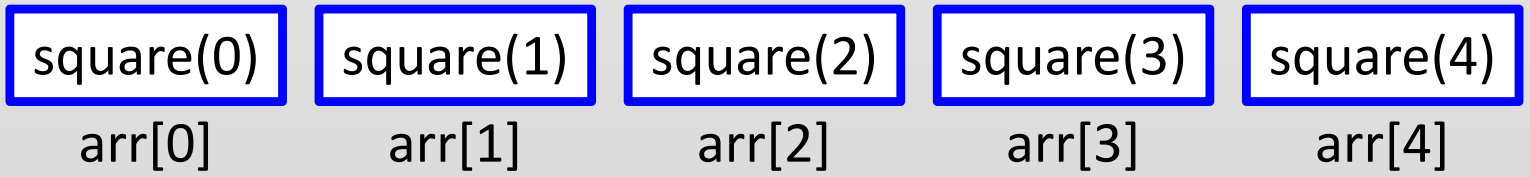
Which of the following expressions returns the area of the square class object? Answer:

```
s.getArea()
```

Evaluate the result of the following coding.

```
square arr[5];
double t = 0;
for (int i=0; i<5; i++){
    arr[i] = square(i);
    t += arr[i].getArea();
}
```

The for loop initialize the array of squares:



If we sum up the return values of the `getArea()` function:

The result is just sum of squares, giving 30.

Q6 Consider a class fraction below.

```
class fraction{
public:
    int num, den;
    fraction(int p, int q){
        num = p;
        den = q;
    }
    fraction operator *(fraction r){
        int x = num*r.num;
        int y = den*r.den;
        return fraction(x,y);
    }
    void print() {
        cout << num << "/" << den << endl;
    }
};
```

If the following lines are executed, what is the output?

```
fraction r1(3,4);
fraction r2(5,6);
fraction r3 = r1*r2;
r3.print();
```

In this question, we overload the binary operator * and applies it on two `fraction` objects. The result is a new `fraction` object.

The num and den of the first object are 3, 4 respectively.
The r.num and r.den of the second object r are 5, 6 respectively.

Therefore according to the function, we have x = 15, y = 24.
Using these values, we form a new `fraction` object as return.
When it is printed out, it gives "15/24".

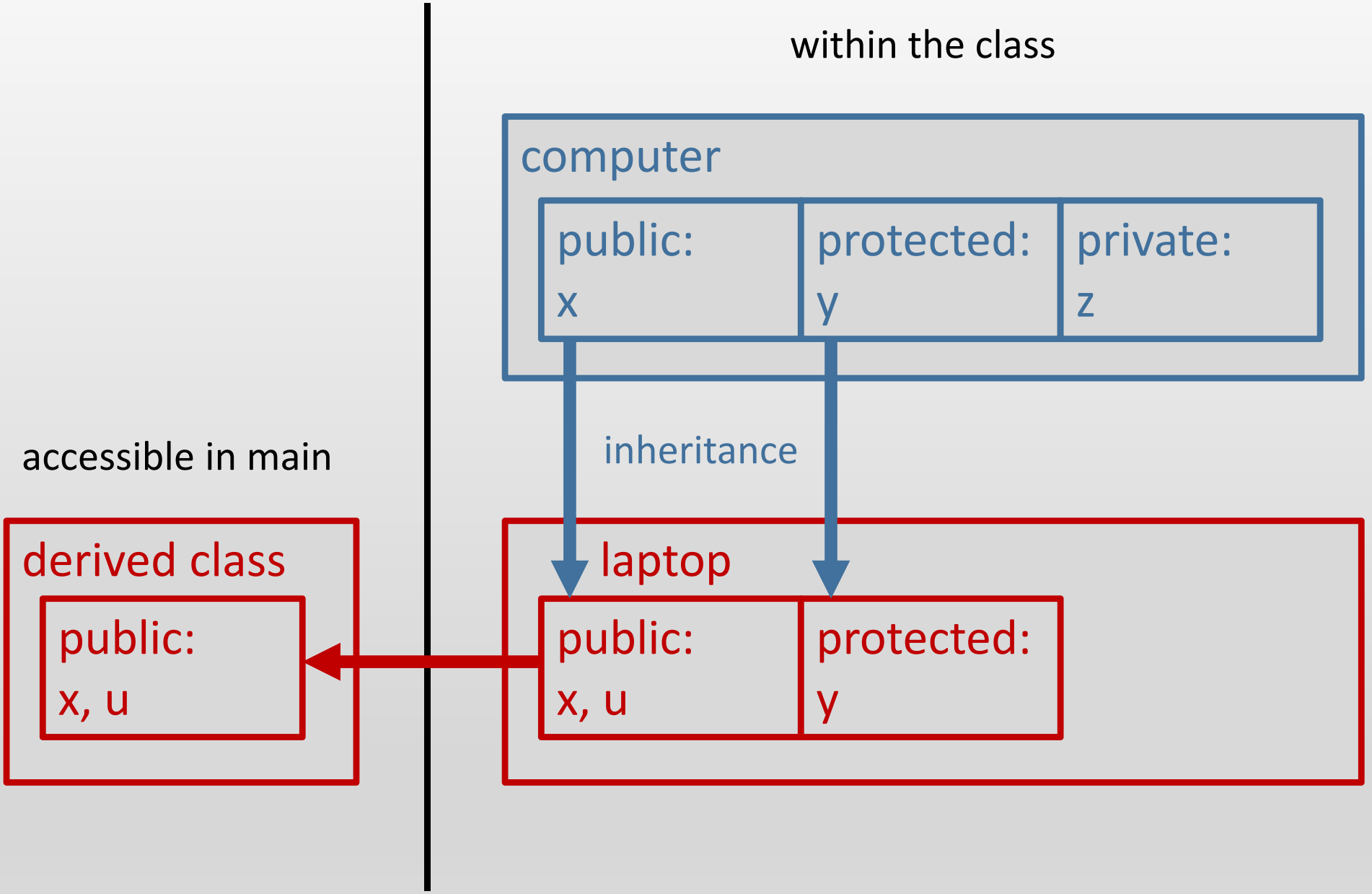
Q7 Consider a class computer and its derived class laptop defined below:

```
class computer{
public:      int x = 1;
private:     int y = 2;
protected:     int z = 3;
};
class laptop: public computer{
public:          int u = 4;
};
```

In the main program, suppose we declare a laptop class object:

```
laptop mylaptop;
```

Which of the following is/are accessible in the main program?

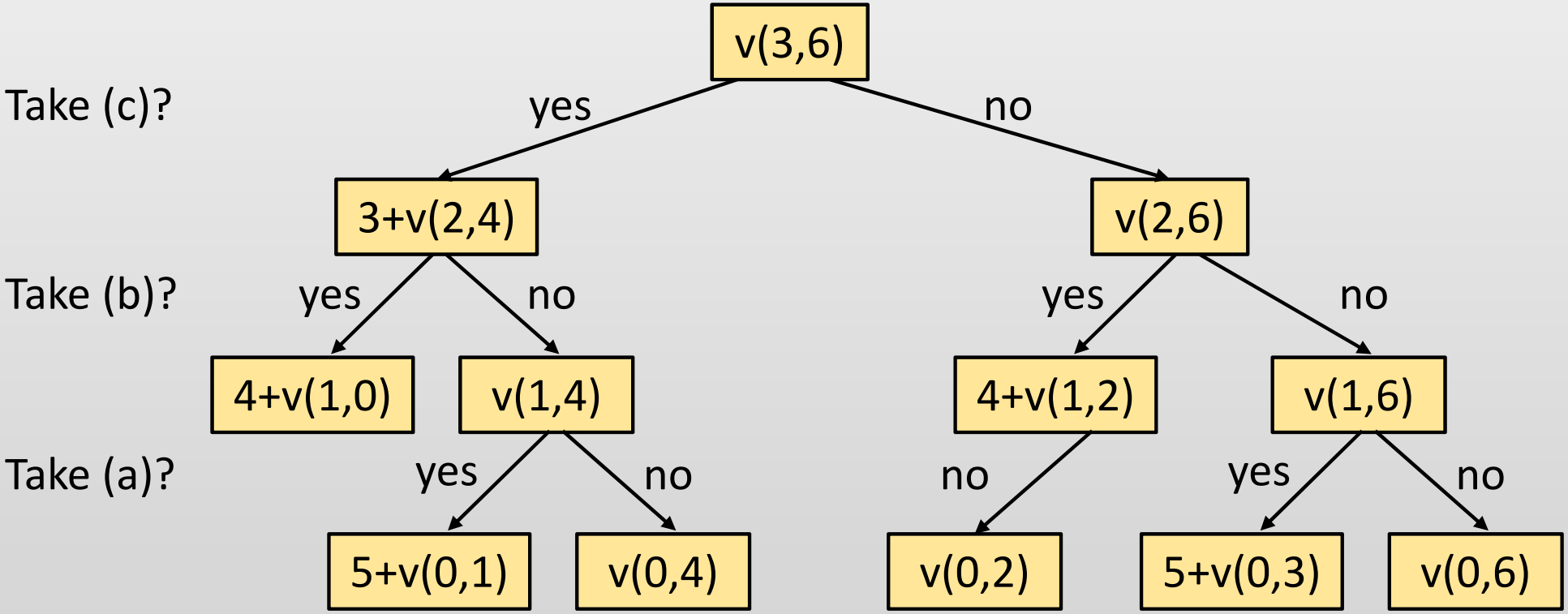

within the class
computer
public:
x
protected:
y
private:
z
inheritance
accessible in main
derived class
public:
x, u
laptop
public:
x, u
protected:
y

Q7 A thief is going to steal some items from a shop. The price and volume are shown below:

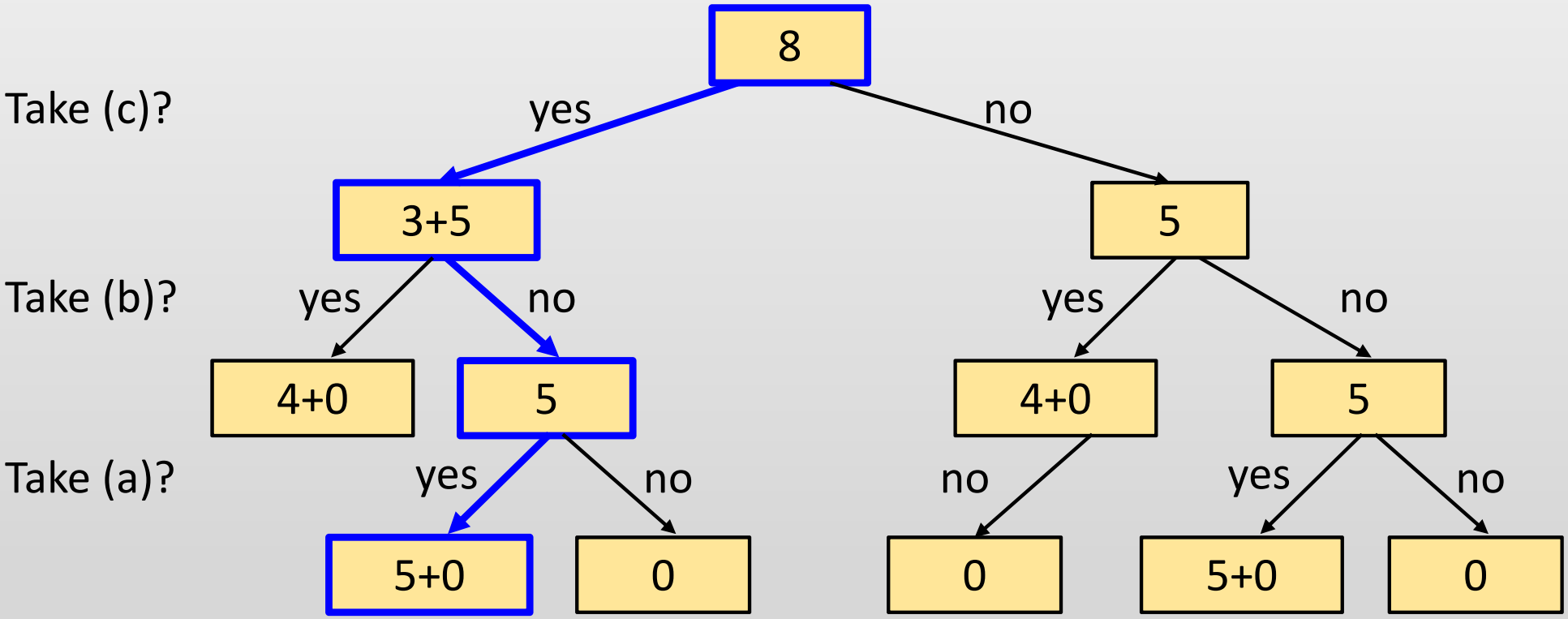
(a) apple - price: 5, volume: 3
(b) banana - price: 4, volume: 4
(c) chocolate - price: 3, volume: 2

The thief set up a function `v(n,t)` based on dynamic programming which evaluates the maximum total price given n items remaining and within (<=) the maximum total volume t allowed. Evaluate `v(3,6)`.

We can consider a decision making process from the very last item (c). If we take the item (yes), we can acquire the price but reduce in remaining capacity.



If we are out of remaining items or capacity, the value function gives 0. If we have two possible branches, take the one with a maximum value. The top most value is 8 resulted from (c),(a).



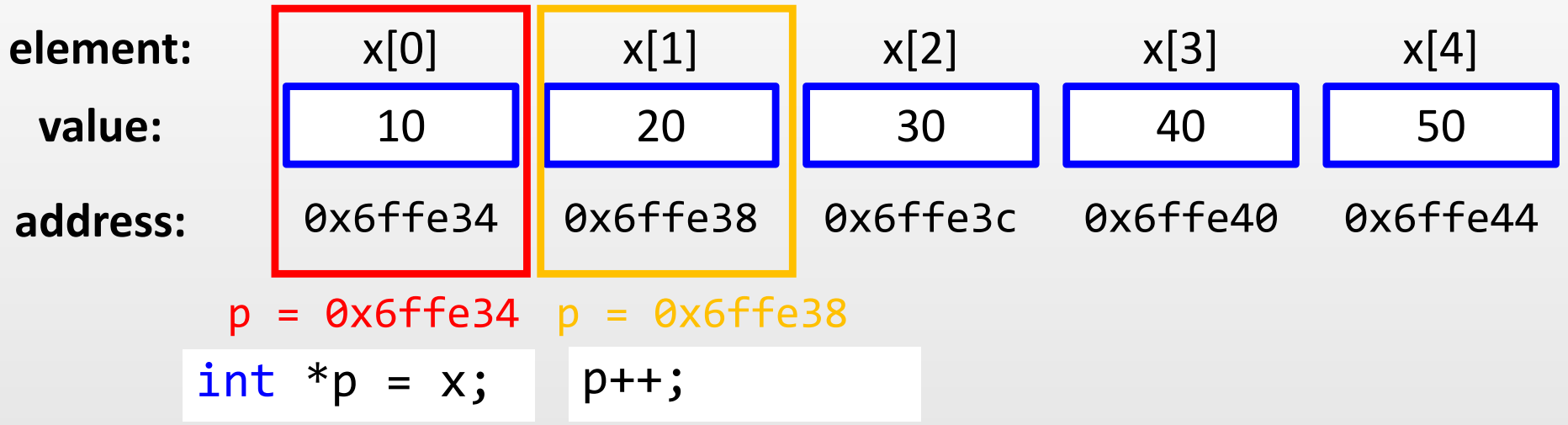
(past quiz) Evaluate the output of the following C++ statements:

```
int x[5] = {10, 20, 30, 40, 50};

int *p = x;
p++;
cout << *p << endl;

int *q = &x[2];
(*q)--;
cout << *q << endl;
```

Many students are confused that they can't distinguish a pointer (a hexadecimal memory address) and its pointed value (e.g., an integer variable). Also they don't understand the relationship between pointer and array.

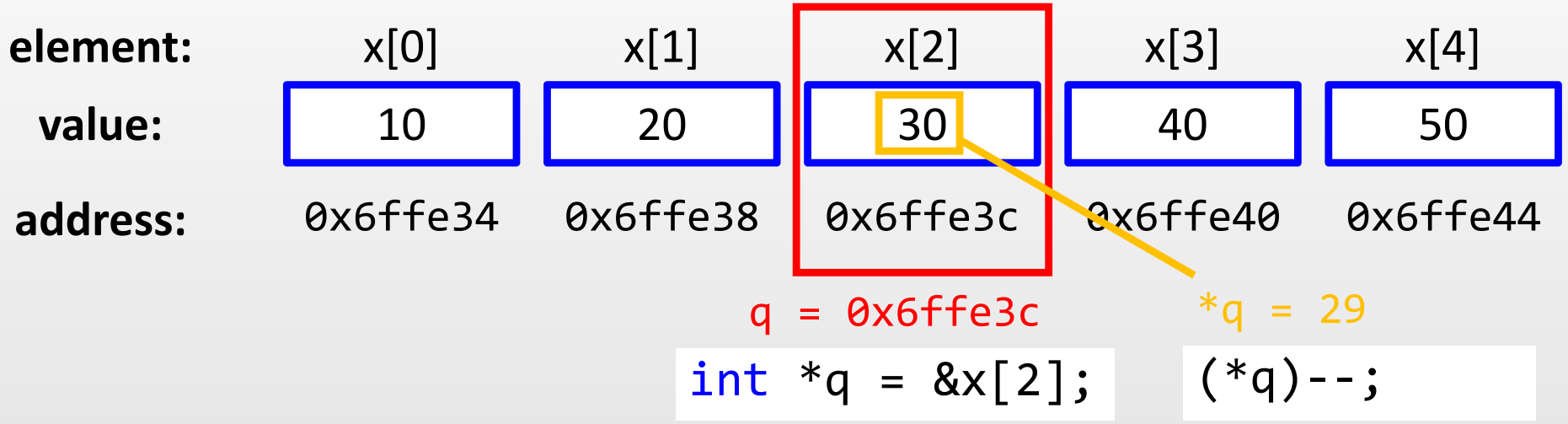


p is an integer pointer, which is initially assigned as the memory address of the whole array x.

When p is incremented, it is updated to the next memory address by adding 4 (bytes) which is the byte size of int data type.

```
cout << *p << endl;
```

When the pointed value *p is printed, it will give 20.



q is an integer pointer, which is initially assigned as the memory address of an element x[2] in the array.

When the pointed value *q is decremented, the corresponding value of the array element x[2] is decremented by 1.

```
cout << *q << endl;
```

When the pointed value *p is printed, it will give 29.

(past exam) Write a class called transaction that represents a transaction of buying some shares of a stock. It should contain the following information in the data field:

code - a code representing the company (string)
shares - number of shares (int)
price - price per share in USD (double)

Under this class, the following should be included:

- a default constructor to create a transaction object with "" for string and 0 for numbers;
- a constructor to create a transaction object giving all required data;
- a function called amount that returns the total amount worth of the transaction;
- a function called print that displays details of the transaction;
- an operator + between two transaction objects that returns a new transaction object with total number of shares and a weighted mean of the prices, given they have the same code. Otherwise return a default transaction object.

You are only required to write the class. You may put both the data field and functions in the public section. The main program is provided to you.

```
#include<iostream>
#include<string>
using namespace std;
//insert the class
int main(){
transaction t1("META", 100, 241.5);
transaction t2("META", 300, 121.5);
transaction t3 = t1 + t2;
t3.print();
return 0;
}
```

The result of this program is shown below:

```
You have bought 400 shares of META at $151.5
The total amount is $60600
```

You need to construct the class part by part. The most essential structure is the class name and sections.

Inside the section (public without any specification, private if data encapsulation is required), include the data field which is the class variables described by the question.

```
class transaction {
public:
// data field
string code;
int shares;
double price;
// other functions
};
```

We will then insert the other functions inside the class.

Constructor is a special kind of function having the same return type as the class. The purpose of a constructor is to declare and initialize an object of this class.
The default constructor doesn't contain input parameters. Object declared in this way will be initialized with the default values.

```
transaction() {
	code = "";
	shares = 0;
	price = 0;
}
```

The parametrized constructor contains input parameters. Those will be used to initialize the class variables of the object.

```
transaction(string s, int n, double p) {
	code = s;
	shares = n;
	price = p;
}
```

Two other functions are declared within the class. The function with a return value should begin with the return value type. The input parameters are optional. Within a class function, we can directly use the class variables for calculation.

```
double amount() {
	return shares * price;
}
```

Notice that the print function doesn't give any return value, so it should begin with void. Following the given format, we can display information about the class object using cout.

```
void print() {
	cout << "You have bought " << shares;
	cout << " shares of " << code;
	cout << " at $" << price << endl;
	cout << "The total amount is $" << amount();
}
```

The meaning of "+" between two transaction object was undefined originally. We can define it ourselves using operator overloading. It should involve three elements:

- Return type (which can be a primitive data type, or a class)
- Operator
- Input parameters (another variable or object).

Here we will need to input another transaction object (call it t). After adding the total shares, evaluating average amount and checking the code, we can return a new transaction object.

```
transaction operator+ (transaction t){
        int n = shares + t.shares;
        double p = (amount() + t.amount()) / n;
        if (code == t.code)
                return transaction(code, n, p);
        else
                return transaction();
}
```