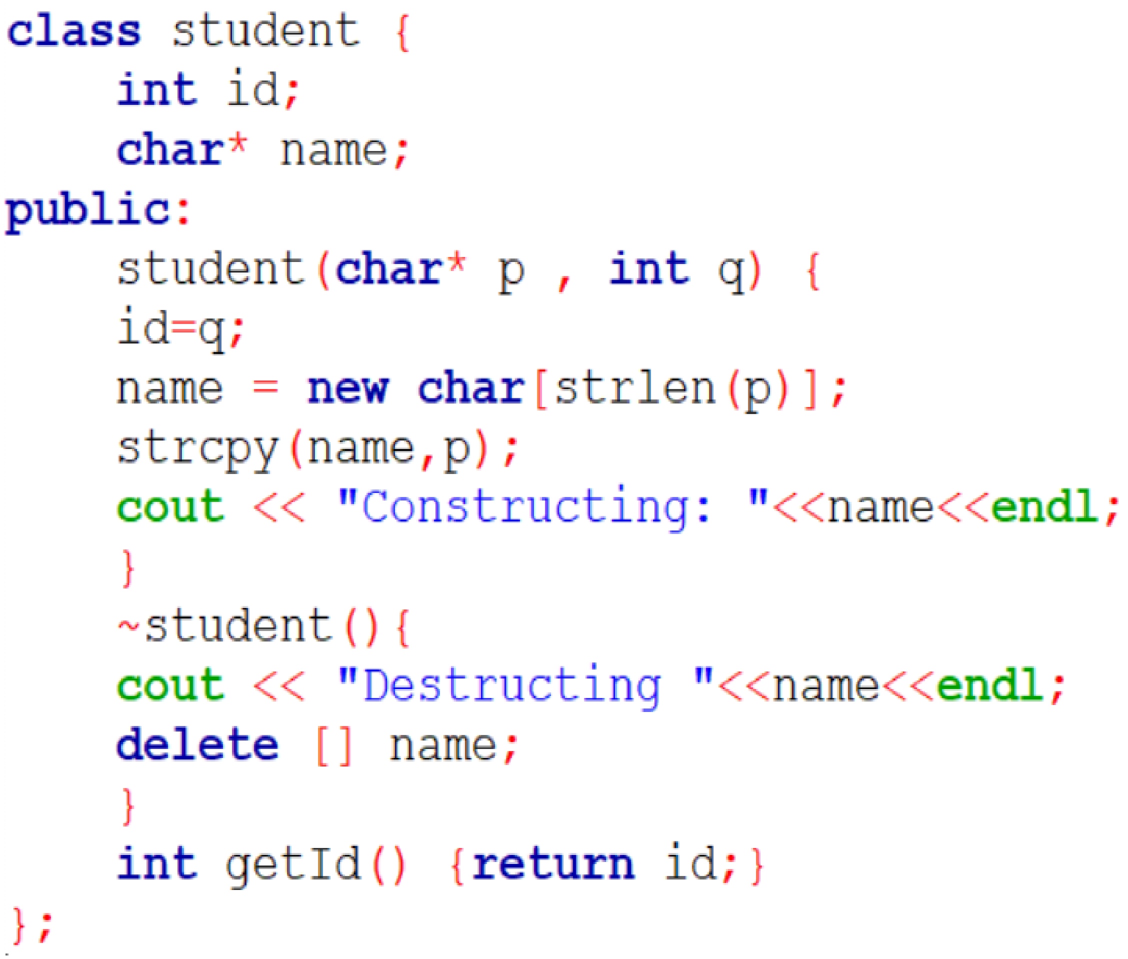
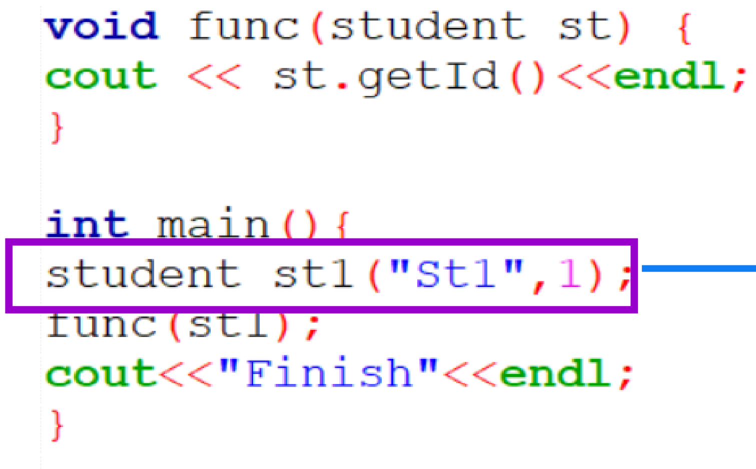
COURSE CODE: CSE 205
COURSE TITLE: OBJECT ORIENTED PROGRAMMING

Prepared by- ***Sumaiya Afroz Mila***

The Old Problem



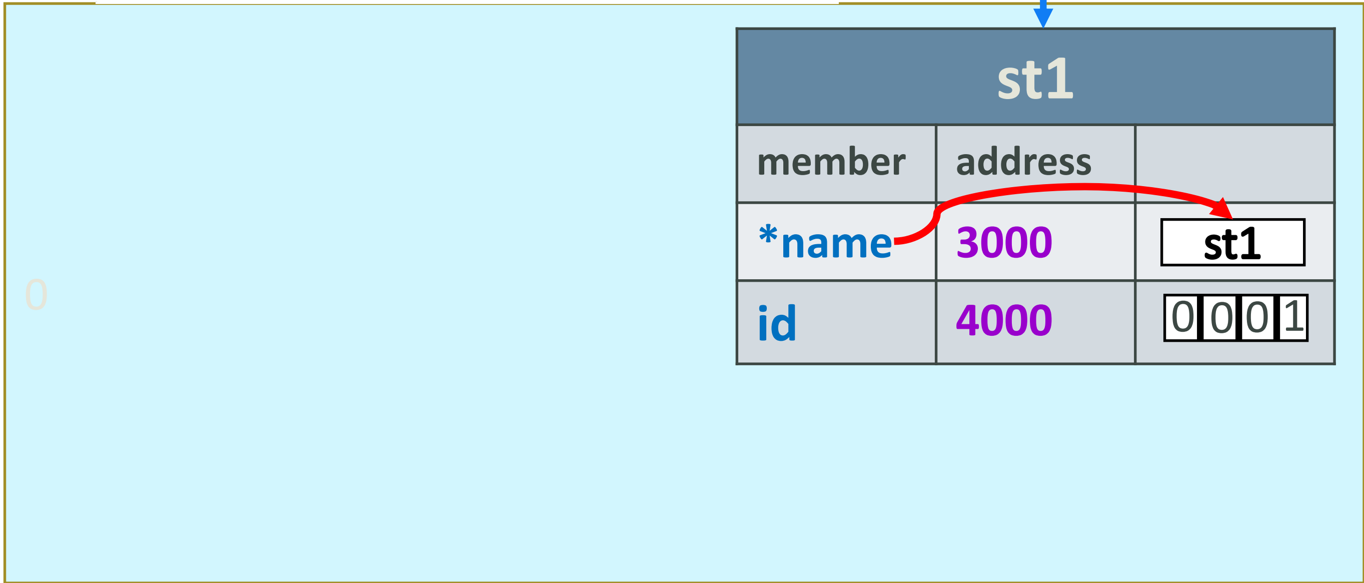
```
class student {
    int id;
    char* name;
public:
    student(char* p , int q) {
    id=q;
    name = new char[strlen(p)];
    strcpy(name,p);
    cout << "Constructing: "<<name<<endl;
    }
    ~student(){
    cout << "Destructing "<<name<<endl;
    delete [] name;
    }
    int getId() {return id;}
};
```



```
void func(student st) {
cout << st.getId()<<endl;
}

int main(){
student st1("St1",1);
func(st1);
cout<<"Finish"<<endl;
}
```

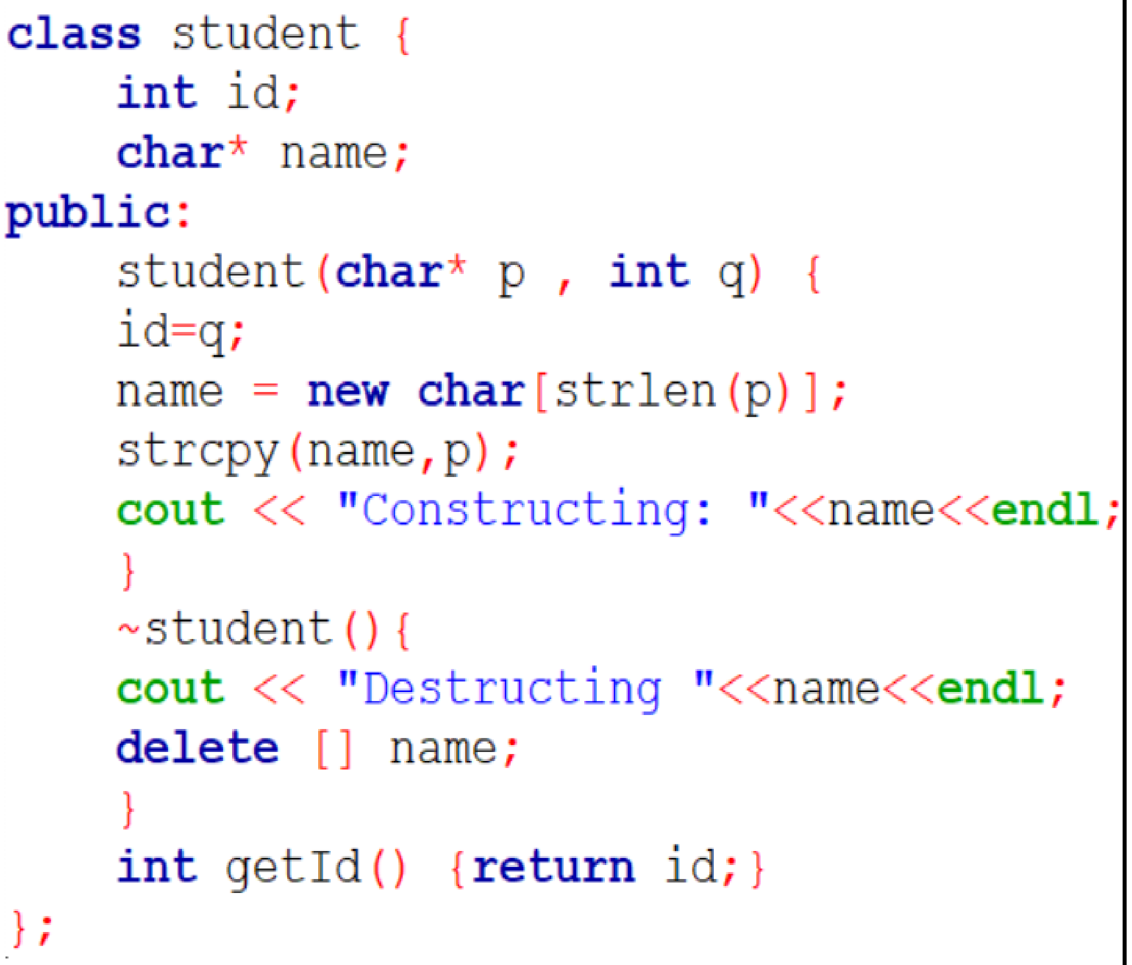
OUTPUT??

Constructing st1

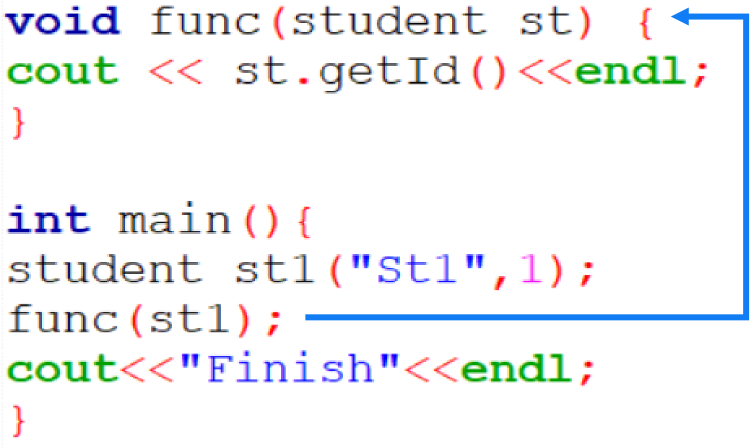


st1		
member	address	
*name	3000	st1
id	4000	0 0 0 1

The Old Problem



```
class student {
    int id;
    char* name;
public:
    student(char* p , int q) {
    id=q;
    name = new char[strlen(p)];
    strcpy(name,p);
    cout << "Constructing: "<<name<<endl;
    }
    ~student(){
    cout << "Destructing "<<name<<endl;
    delete [] name;
    }
    int getId() {return id;}
};
```

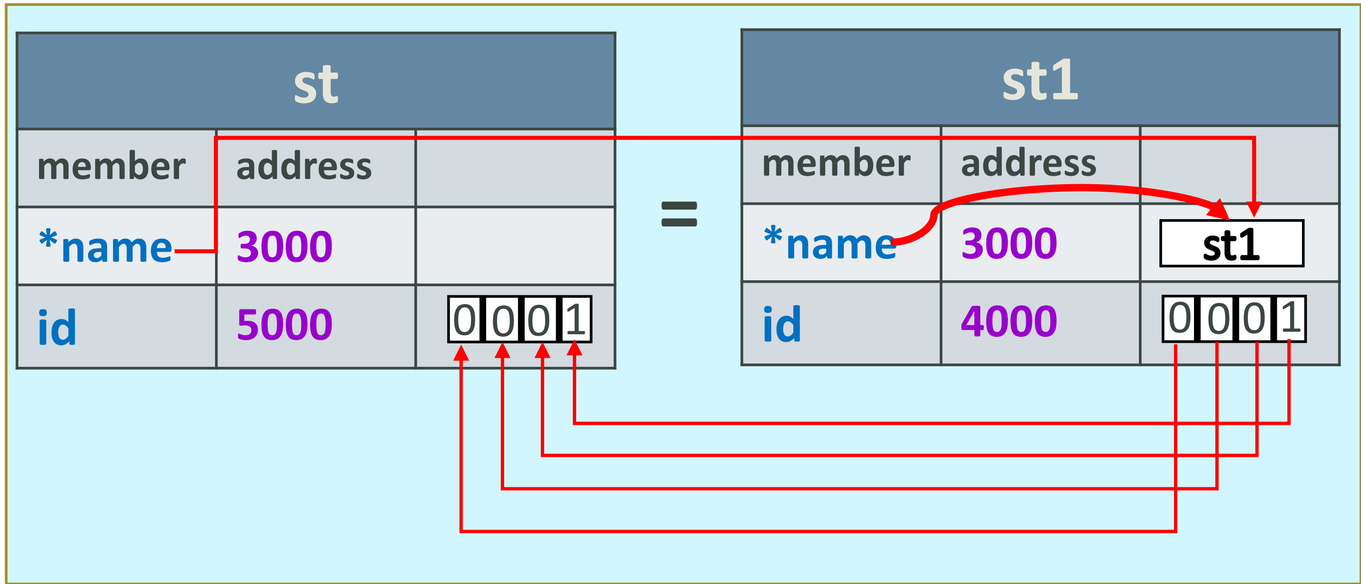


```
void func(student st) {
cout << st.getId()<<endl;
}

int main(){
student st1("St1",1);
func(st1);
cout<<"Finish"<<endl;
}
```

OUTPUT??

Constructing st1

st = st1(bitwise copy)



st		
member	address	
*name	3000	
id	5000	0 0 0 1

=

st1		
member	address	
*name	3000	st1
id	4000	0 0 0 1

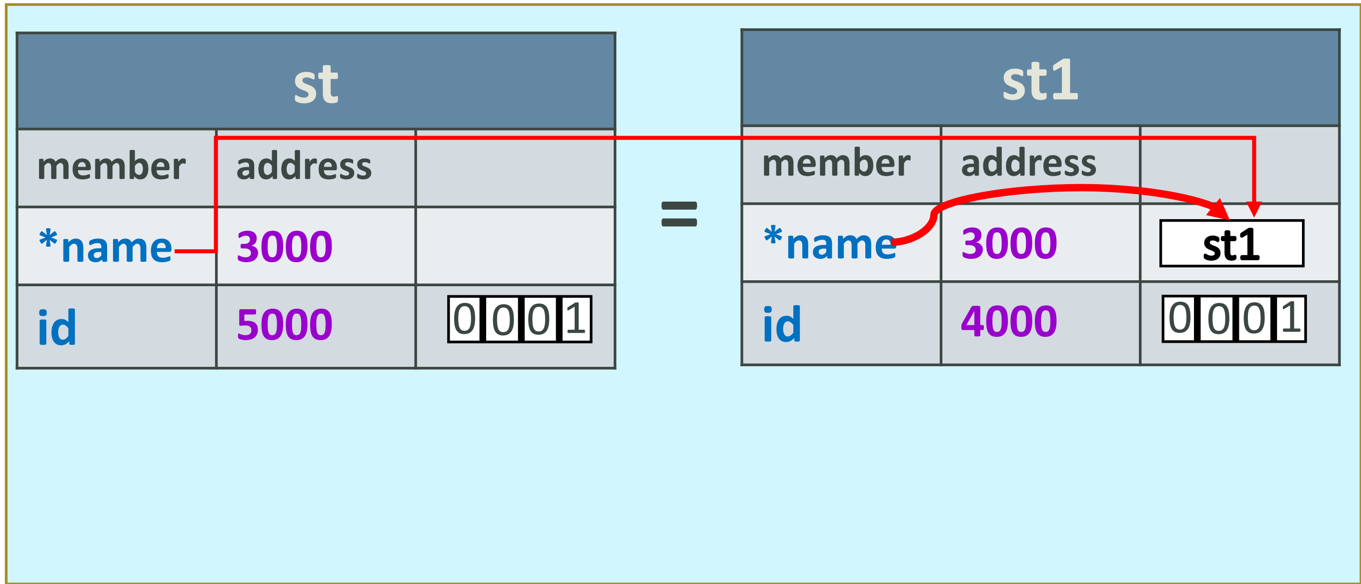
The Old Problem

```cpp
class student {
    int id;
    char* name;
public:
    student(char* p , int q) {
    id=q;
    name = new char[strlen(p)];
    strcpy(name,p);
    cout << "Constructing: "<<name<<endl;
    }
    ~student(){
    cout << "Destructing "<<name<<endl;
    delete [] name;
    }
    int getId() {return id;}
};
```

```cpp
void func(student st) {
cout << st.getId()<<endl;
}

int main(){
student st1("St1",1);
func(st1);
cout<<"Finish"<<endl;
}
```

OUTPUT??

Constructing st1
1



st		
member	address	
*name	3000	
id	5000	0 0 0 1

=

st1		
member	address	
*name	3000	st1
id	4000	0 0 0 1

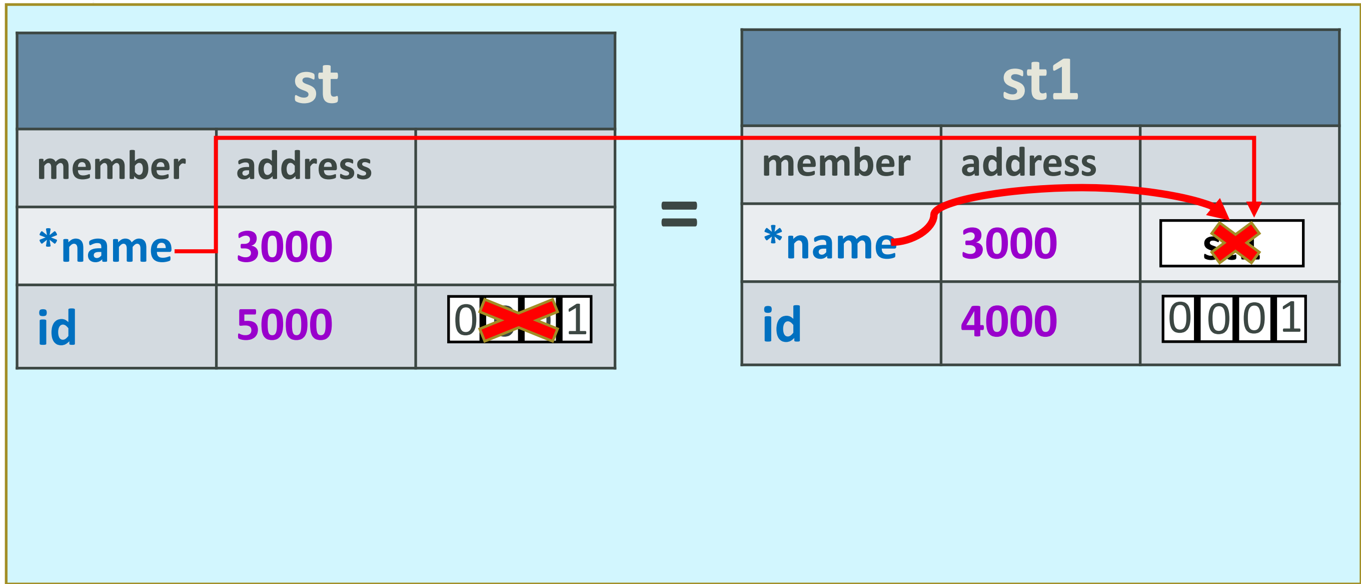
The Old Problem

```
class student {
     int id;
     char* name;
public:
     student(char* p , int q) {
     id=q;
     name = new char[strlen(p)];
     strcpy(name,p);
     cout << "Constructing: "<<name<<endl;
     }
     ~student(){
     cout << "Destructing "<<name<<endl;
     delete [] name;
     }
     int getId() {return id;}
};
```

```
void func(student st) {
cout << st.getId()<<endl;
}

int main(){
student st1("St1",1);
func(st1);
cout<<"Finish"<<endl;
}
```

OUTPUT??

```
Constructing st1
1
Destructing st1
```



st		
member	address	
*name	3000	
id	5000	0 0 0 1

=

st1		
member	address	
*name	3000	St1
id	4000	0 0 0 1

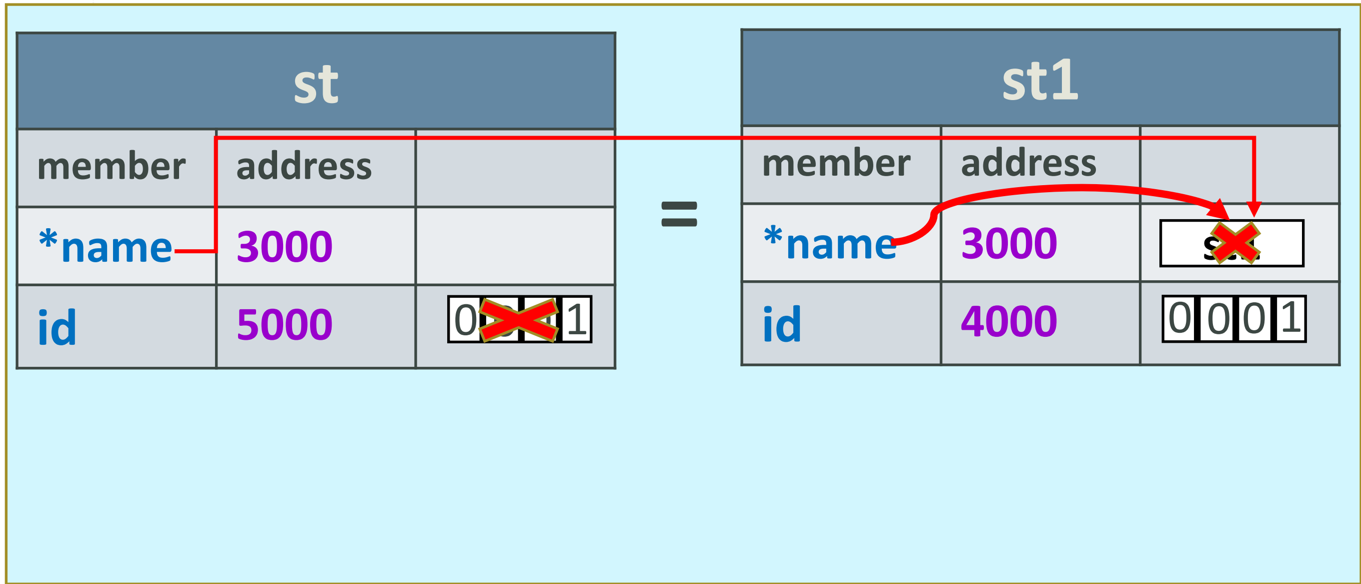
The Old Problem

```
class student {
    int id;
    char* name;
public:
    student(char* p , int q) {
    id=q;
    name = new char[strlen(p)];
    strcpy(name,p);
    cout << "Constructing: "<<name<<endl;
    }
    ~student(){
    cout << "Destructing "<<name<<endl;
    delete [] name;
    }
    int getId() {return id;}
};
```

```
void func(student st) {
cout << st.getId()<<endl;
}

int main(){
student st1("St1",1);
func(st1);
cout<<"Finish"<<endl;
}
```

OUTPUT??

```
Constructing st1
1
Destructing st1
Finish
```



st		
member	address	
*name	3000	
id	5000	0 ~~0 0~~ 1

=

st1		
member	address	
*name	3000	~~st1~~
id	4000	0 0 0 1

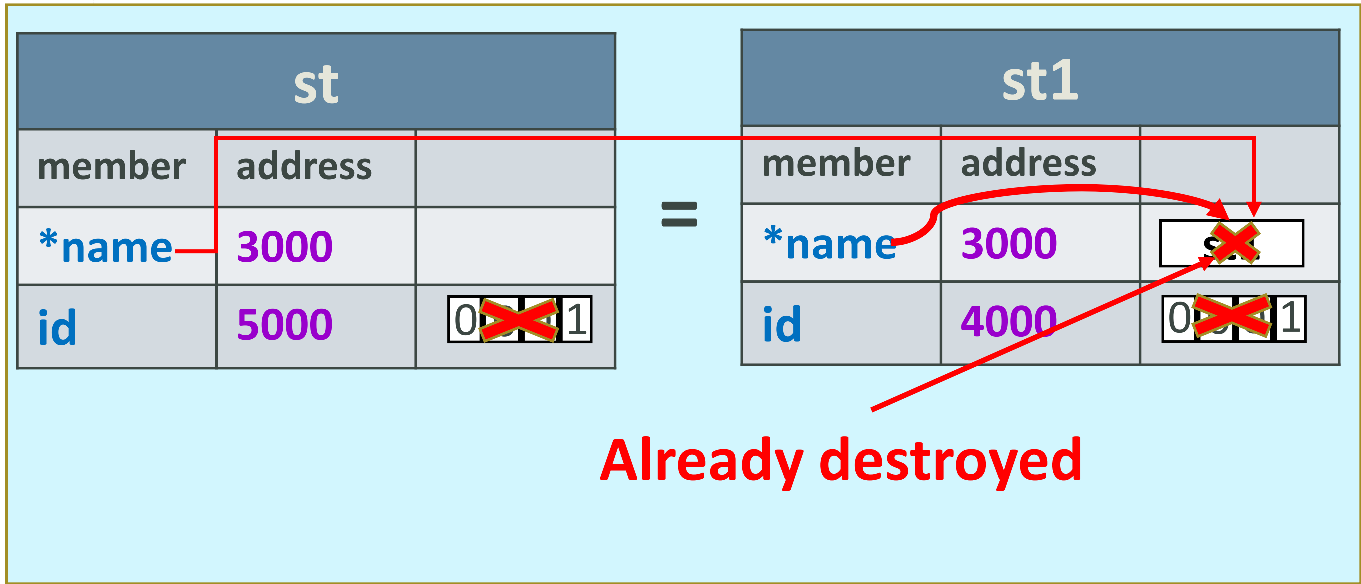
The Old Problem

```
class student {
    int id;
    char* name;
public:
    student(char* p , int q) {
    id=q;
    name = new char[strlen(p)];
    strcpy(name,p);
    cout << "Constructing: "<<name<<endl;
    }
    ~student(){
    cout << "Destructing "<<name<<endl;
    delete [] name;
    }
    int getId() {return id;}
};
```

```
void func(student st) {
cout << st.getId()<<endl;
}

int main(){
student st1("St1",1);
func(st1);
cout<<"Finish"<<endl;
}
```

OUTPUT??

```
Constructing st1
1
Destructing st1
Finish
Destructing st1
```



st		
member	address	
*name	3000	
id	5000	0 ~~0 0~~ 1

=

st1		
member	address	
*name	3000	~~St1~~
id	4000	0 ~~0 0~~ 1

Already destroyed

A New Solution – User Defined Copy Constructor

- The ***default copy constructor*** makes a bitwise copy.
- Copy Constructor invoked in 3 ways –
  - When an object is used to **initialize** another in a **declaration** statement(***student st2=st1***;)
  - **When an object is passed as a parameter to a function**
  - **When a temporary object is created for use as a return value by a function**
- ***User defined copy constructor*** specifies exactly what occurs when a copy of an object is made

Copy Constructor Syntax

- Most Common Form –

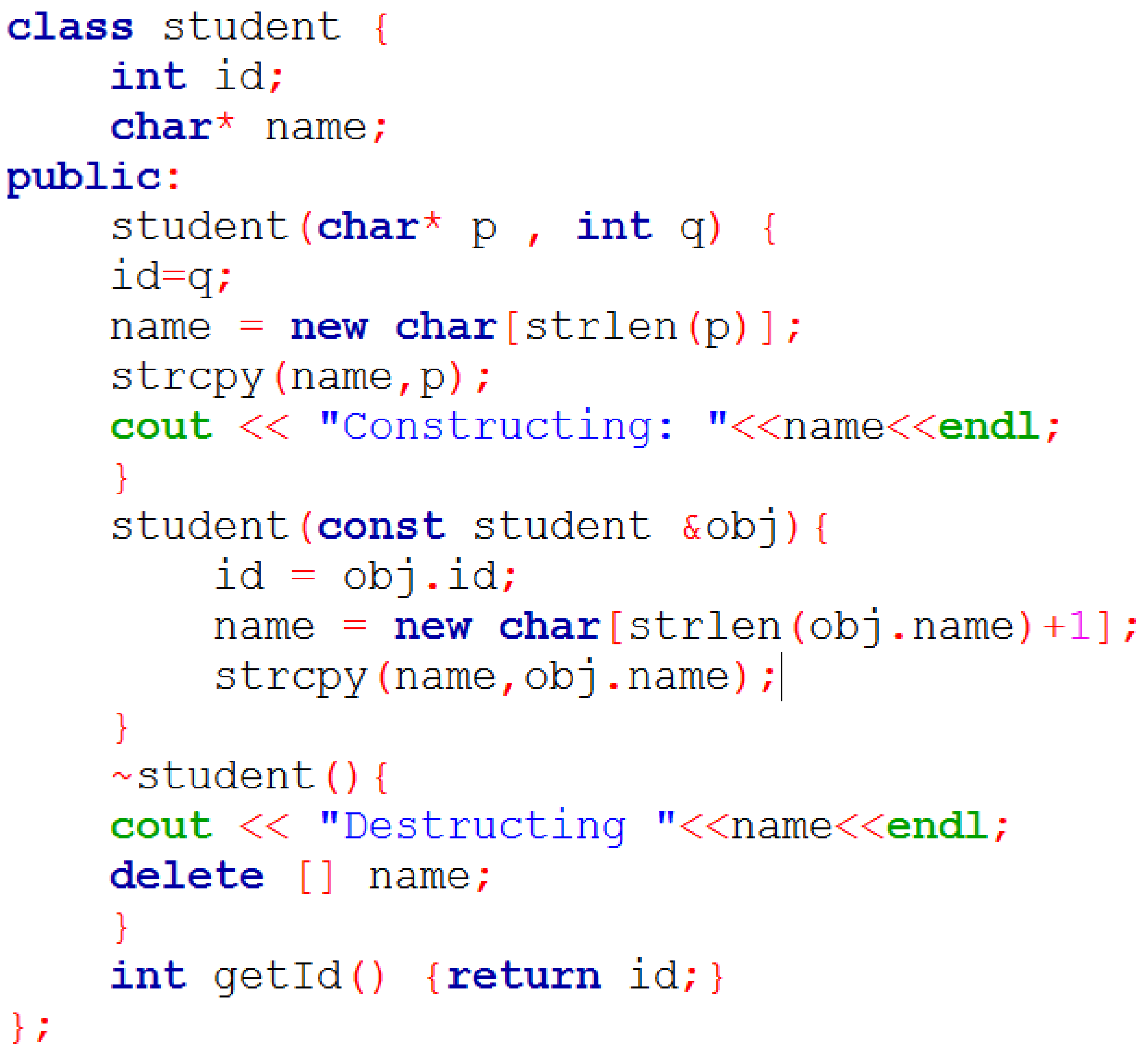
```
classname (const classname &obj){
// body
}
```

Here ***obj*** is a **reference** to an **object** that is being **used to initialize another object.**

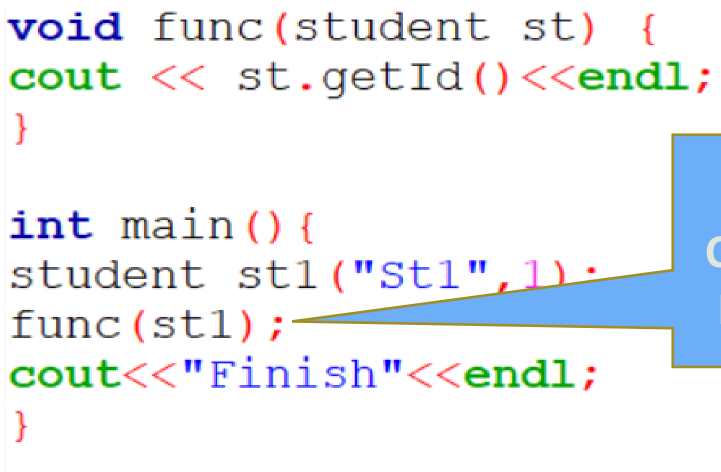
- Copy Constructor of the student class –

```
student (const student &ob){
        id = obj.id;
        name = new char[strlen(obj.name)+1];
        strcpy(name,obj.name);
}
```

Copy Constructor Invocation



```
class student {
    int id;
    char* name;
public:
    student(char* p , int q) {
    id=q;
    name = new char[strlen(p)];
    strcpy(name,p);
    cout << "Constructing: "<<name<<endl;
    }
    student(const student &obj){
        id = obj.id;
        name = new char[strlen(obj.name)+1];
        strcpy(name,obj.name);
    }
    ~student(){
    cout << "Destructing "<<name<<endl;
    delete [] name;
    }
    int getId() {return id;}
};
```



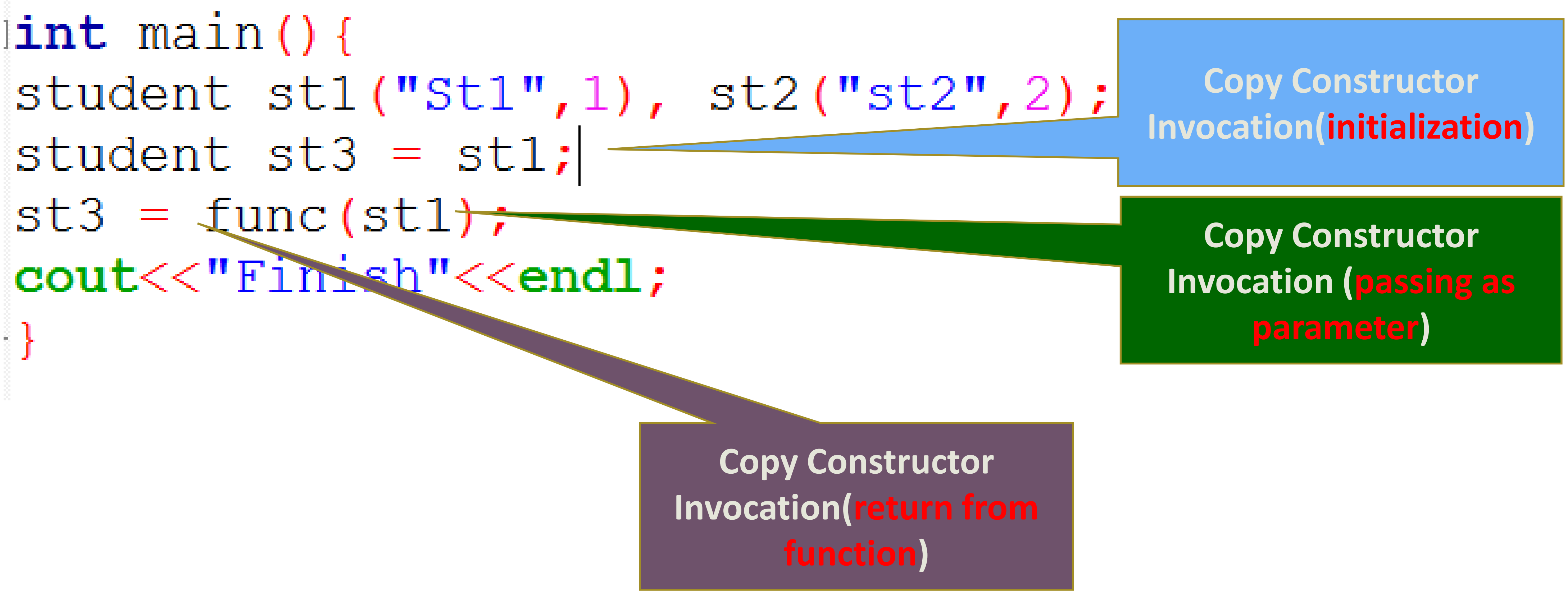
```
void func(student st) {
cout << st.getId()<<endl;
}

int main(){
student st1("St1",1);
func(st1);
cout<<"Finish"<<endl;
}
```

Copy Constructor Invocation

Copy Constructor Invocation

- Student s1 = s2; // y explicitly initializing x
- func(s2); // y passed as a parameter
- s1 = func(s1); // y receiving a returned object

Copy Constructor Invocation Example



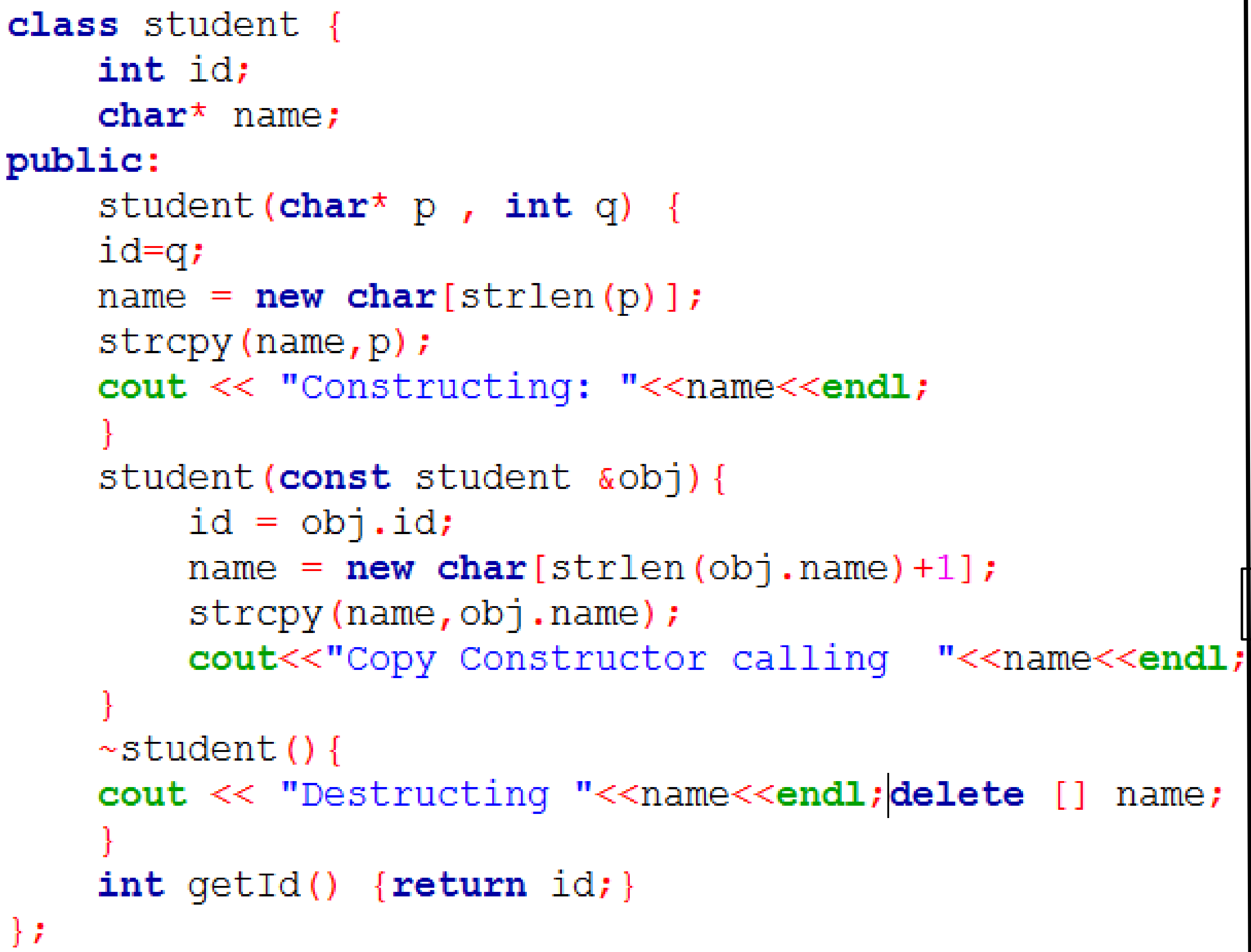
```
int main(){
student st1("St1",1), st2("st2",2);
student st3 = st1;
st3 = func(st1);
cout<<"Finish"<<endl;
}
```

Copy Constructor Invocation(initialization)

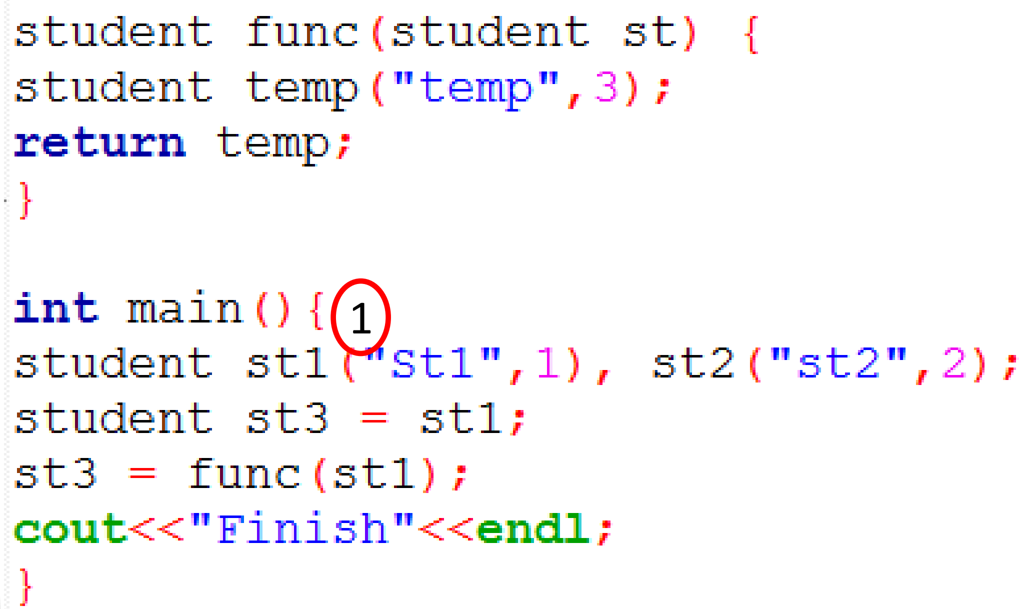
Copy Constructor Invocation (passing as parameter)

Copy Constructor Invocation(return from function)

Copy Constructor Invocation Example



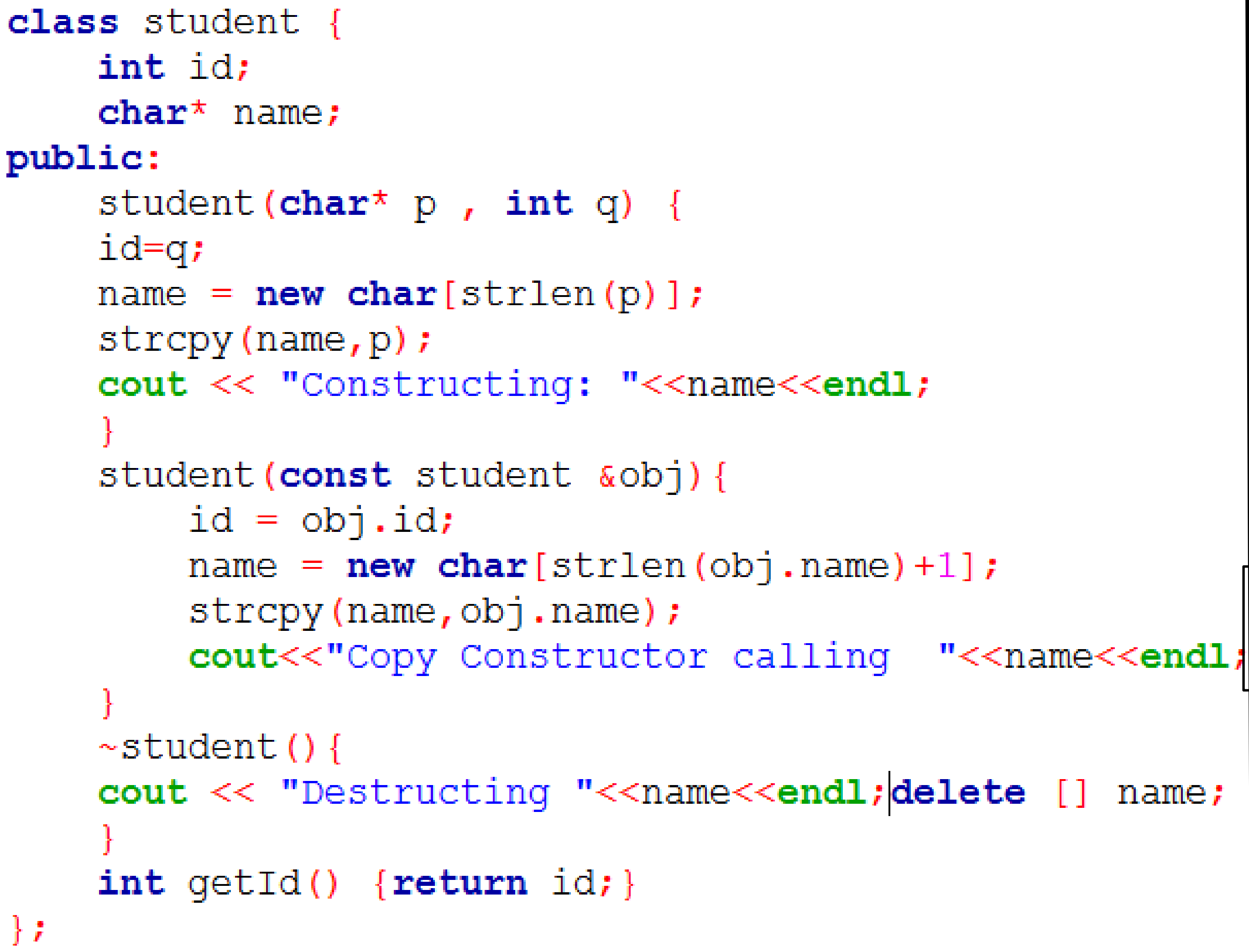
```
class student {
    int id;
    char* name;
public:
    student(char* p , int q) {
    id=q;
    name = new char[strlen(p)];
    strcpy(name,p);
    cout << "Constructing: "<<name<<endl;
    }
    student(const student &obj){
        id = obj.id;
        name = new char[strlen(obj.name)+1];
        strcpy(name,obj.name);
        cout<<"Copy Constructor calling  "<<name<<endl;
    }
    ~student(){
    cout << "Destructing "<<name<<endl;delete [] name;
    }
    int getId() {return id;}
};
```



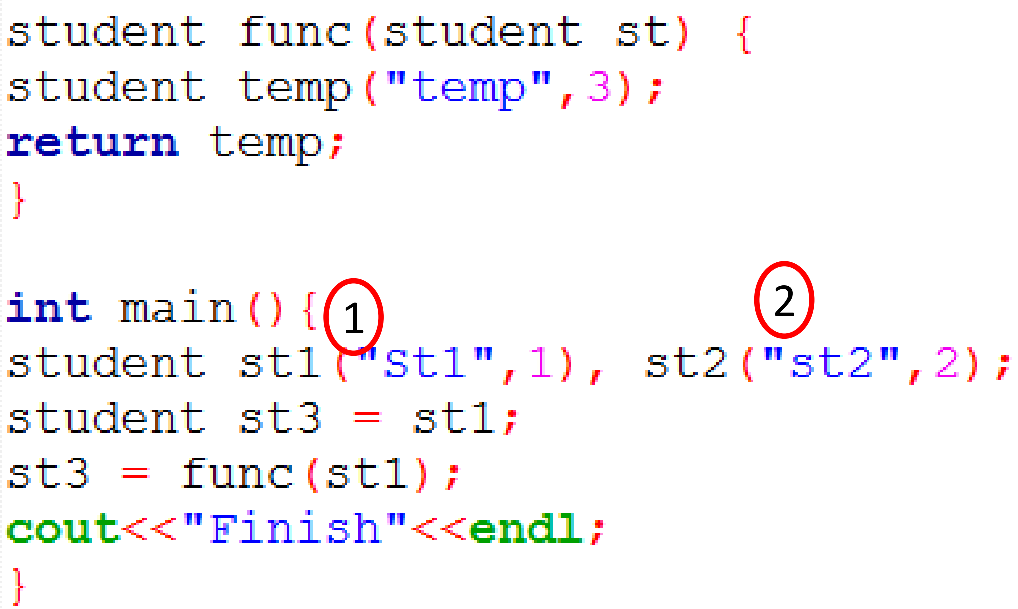
```
student func(student st) {
student temp("temp",3);
return temp;
}

int main(){
student st1("St1",1), st2("st2",2);
student st3 = st1;
st3 = func(st1);
cout<<"Finish"<<endl;
}
```

1.Constructing: St1 (In main)	

Copy Constructor Invocation Example



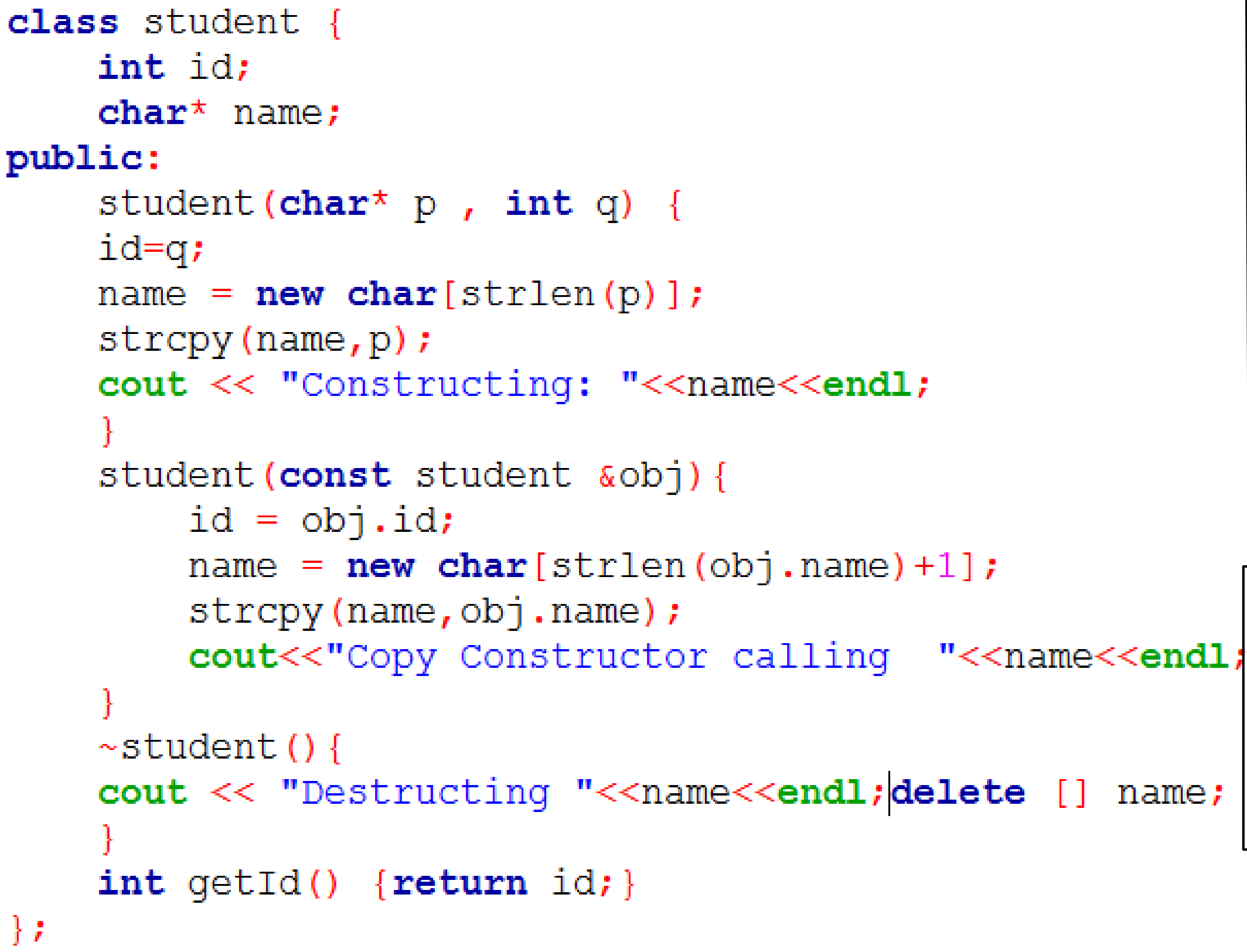
```
class student {
    int id;
    char* name;
public:
    student(char* p , int q) {
    id=q;
    name = new char[strlen(p)];
    strcpy(name,p);
    cout << "Constructing: "<<name<<endl;
    }
    student(const student &obj){
        id = obj.id;
        name = new char[strlen(obj.name)+1];
        strcpy(name,obj.name);
        cout<<"Copy Constructor calling  "<<name<<endl;
    }
    ~student(){
    cout << "Destructing "<<name<<endl;delete [] name;
    }
    int getId() {return id;}
};
```



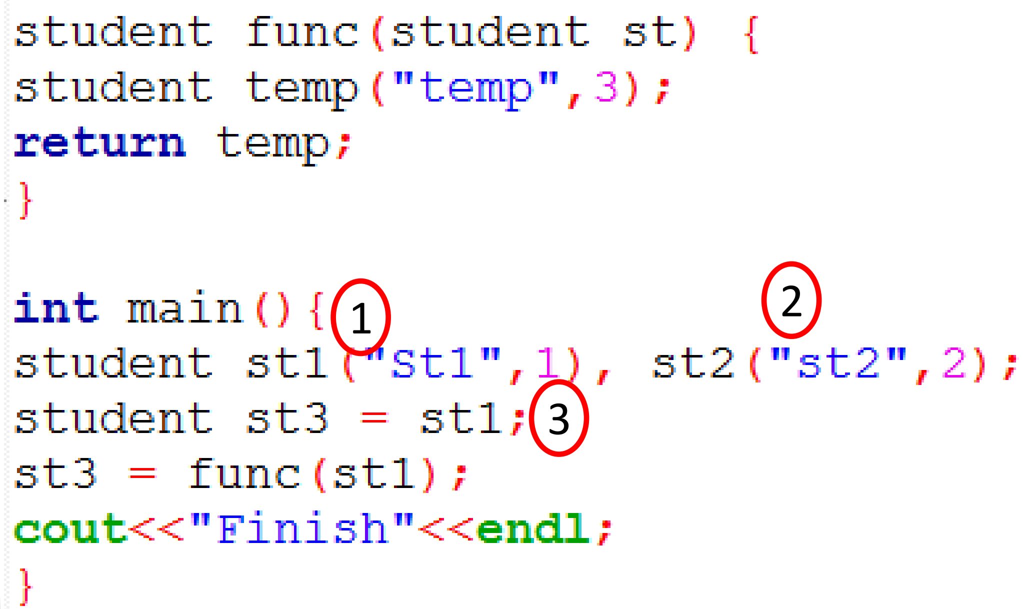
```
student func(student st) {
student temp("temp",3);
return temp;
}

int main(){
student st1("St1",1), st2("st2",2);
student st3 = st1;
st3 = func(st1);
cout<<"Finish"<<endl;
}
```

1.Constructing: St1 (In main)
2.Constructing: St2 (In main)

Copy Constructor Invocation Example



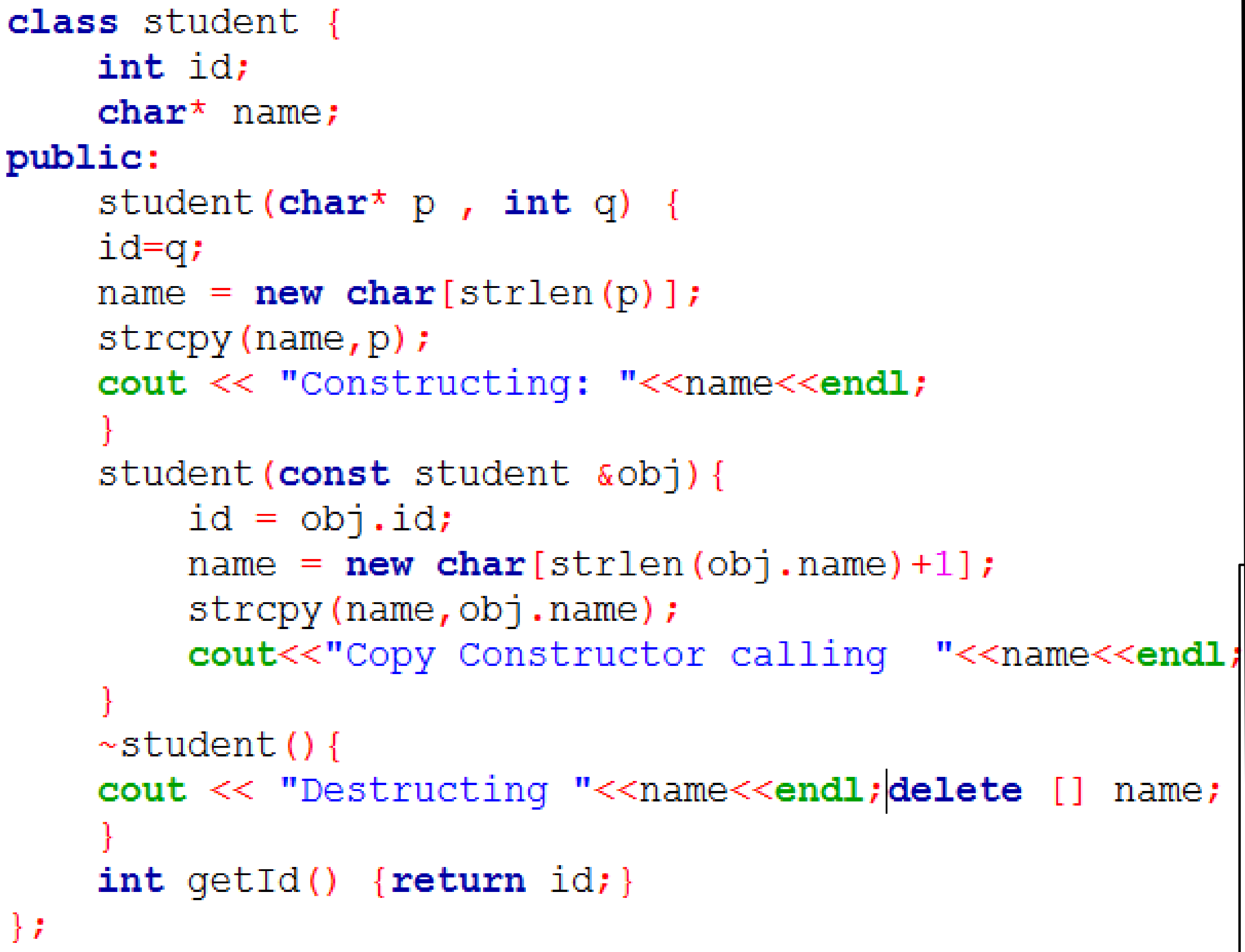
```
class student {
    int id;
    char* name;
public:
    student(char* p , int q) {
    id=q;
    name = new char[strlen(p)];
    strcpy(name,p);
    cout << "Constructing: "<<name<<endl;
    }
    student(const student &obj){
        id = obj.id;
        name = new char[strlen(obj.name)+1];
        strcpy(name,obj.name);
        cout<<"Copy Constructor calling  "<<name<<endl;
    }
    ~student(){
    cout << "Destructing "<<name<<endl;delete [] name;
    }
    int getId() {return id;}
};
```



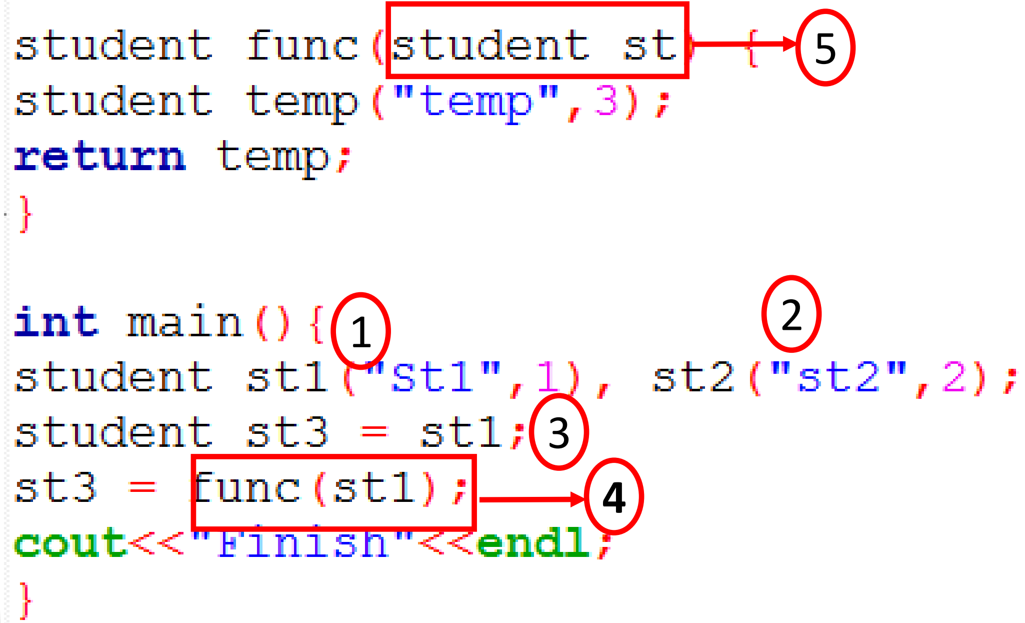
```
student func(student st) {
student temp("temp",3);
return temp;
}

int main(){
student st1("St1",1), st2("st2",2);
student st3 = st1;
st3 = func(st1);
cout<<"Finish"<<endl;
}
```

1.Constructing: St1(In main)
2.Constructing: St2(In main)
3.Copy Constructor Calling St1 (In main)

Copy Constructor Invocation Example



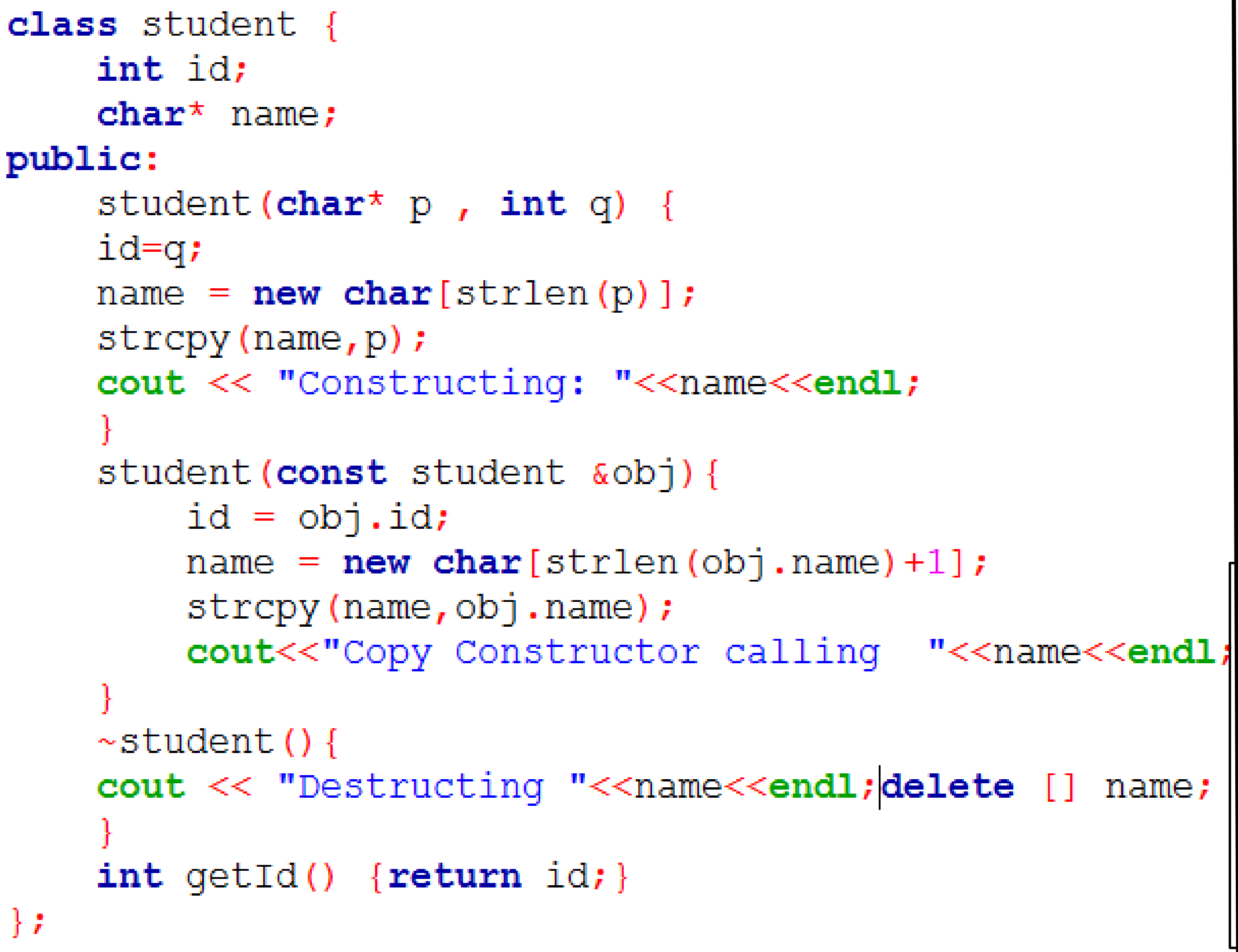
```
class student {
    int id;
    char* name;
public:
    student(char* p , int q) {
    id=q;
    name = new char[strlen(p)];
    strcpy(name,p);
    cout << "Constructing: "<<name<<endl;
    }
    student(const student &obj){
        id = obj.id;
        name = new char[strlen(obj.name)+1];
        strcpy(name,obj.name);
        cout<<"Copy Constructor calling  "<<name<<endl;
    }
    ~student(){
    cout << "Destructing "<<name<<endl;delete [] name;
    }
    int getId() {return id;}
};
```



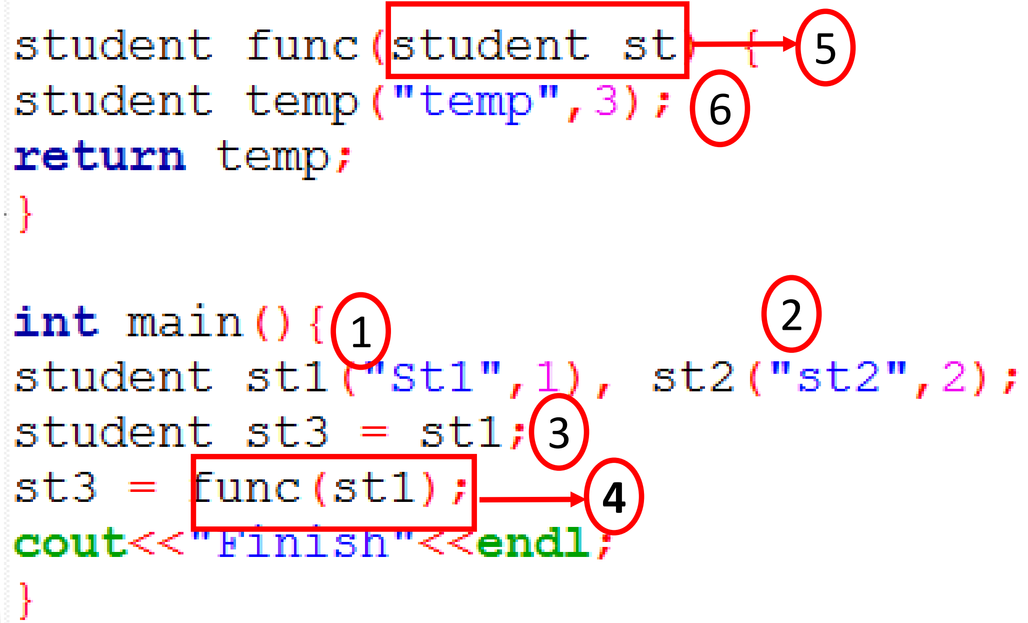
```
student func(student st) {
student temp("temp",3);
return temp;
}

int main(){
student st1("St1",1), st2("st2",2);
student st3 = st1;
st3 = func(st1);
cout<<"Finish"<<endl;
}
```

1.Constructing: St1(In main)
2.Constructing: St2(In main)
3.Copy Constructor Calling St1 (In main)
4.Copy Constructor Calling St1 (In function parameter)

Copy Constructor Invocation Example



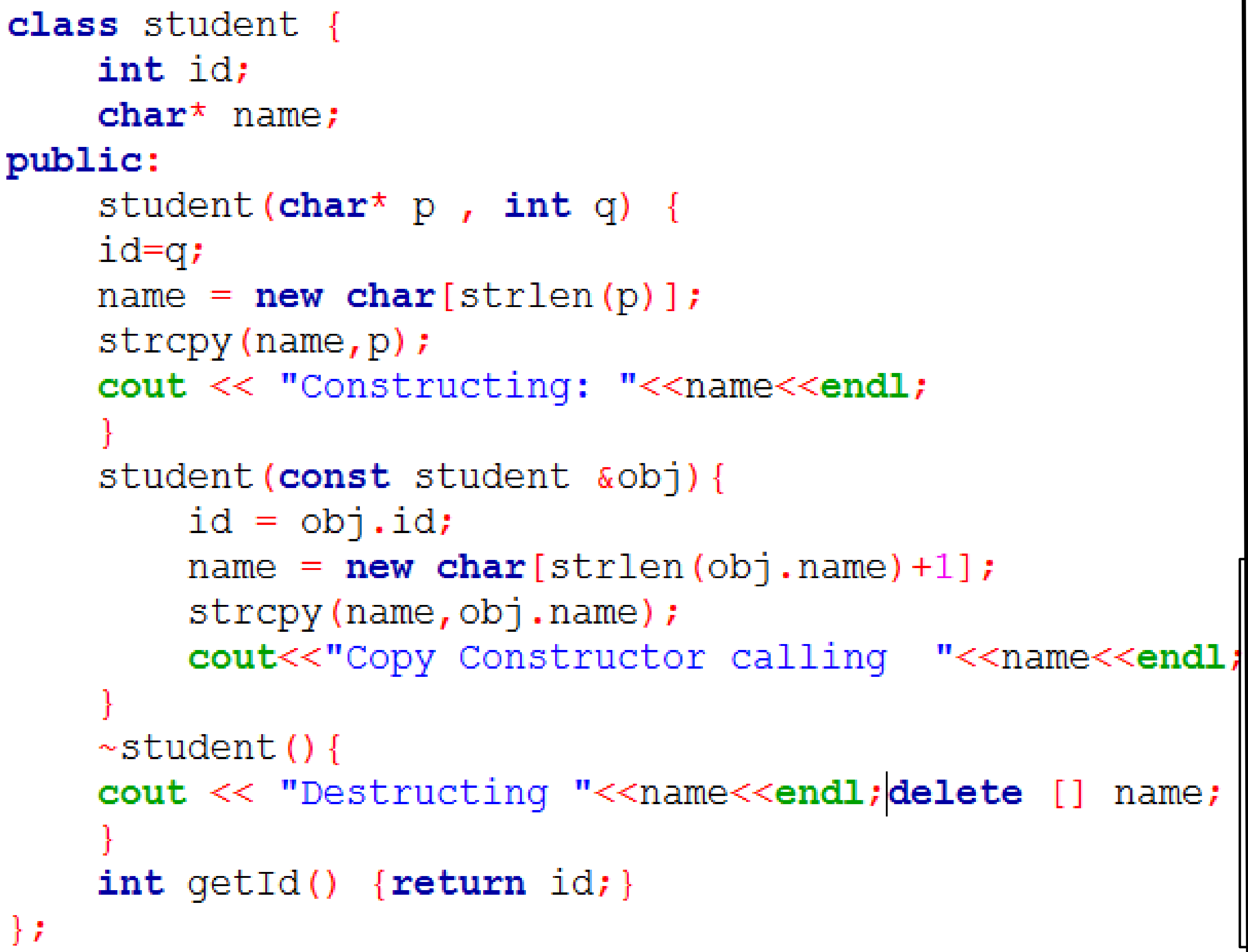
```
class student {
    int id;
    char* name;
public:
    student(char* p , int q) {
    id=q;
    name = new char[strlen(p)];
    strcpy(name,p);
    cout << "Constructing: "<<name<<endl;
    }
    student(const student &obj){
        id = obj.id;
        name = new char[strlen(obj.name)+1];
        strcpy(name,obj.name);
        cout<<"Copy Constructor calling  "<<name<<endl;
    }
    ~student(){
    cout << "Destructing "<<name<<endl;delete [] name;
    }
    int getId() {return id;}
};
```



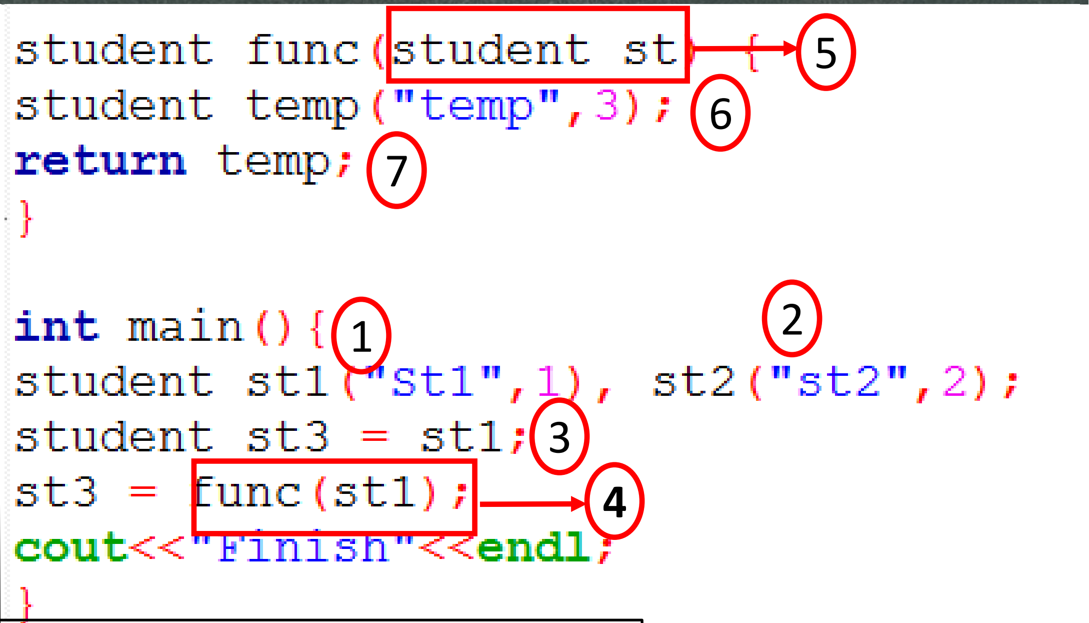
```
student func(student st) {
student temp("temp",3);
return temp;
}

int main(){
student st1("St1",1), st2("st2",2);
student st3 = st1;
st3 = func(st1);
cout<<"Finish"<<endl;
}
```

1.Constructing: St1(In main)
2.Constructing: St2(In main)
3.Copy Constructor Calling St1 (In main)
4.Copy Constructor Calling St1 (In function parameter)
6.Constructing: temp(In local)

Copy Constructor Invocation Example



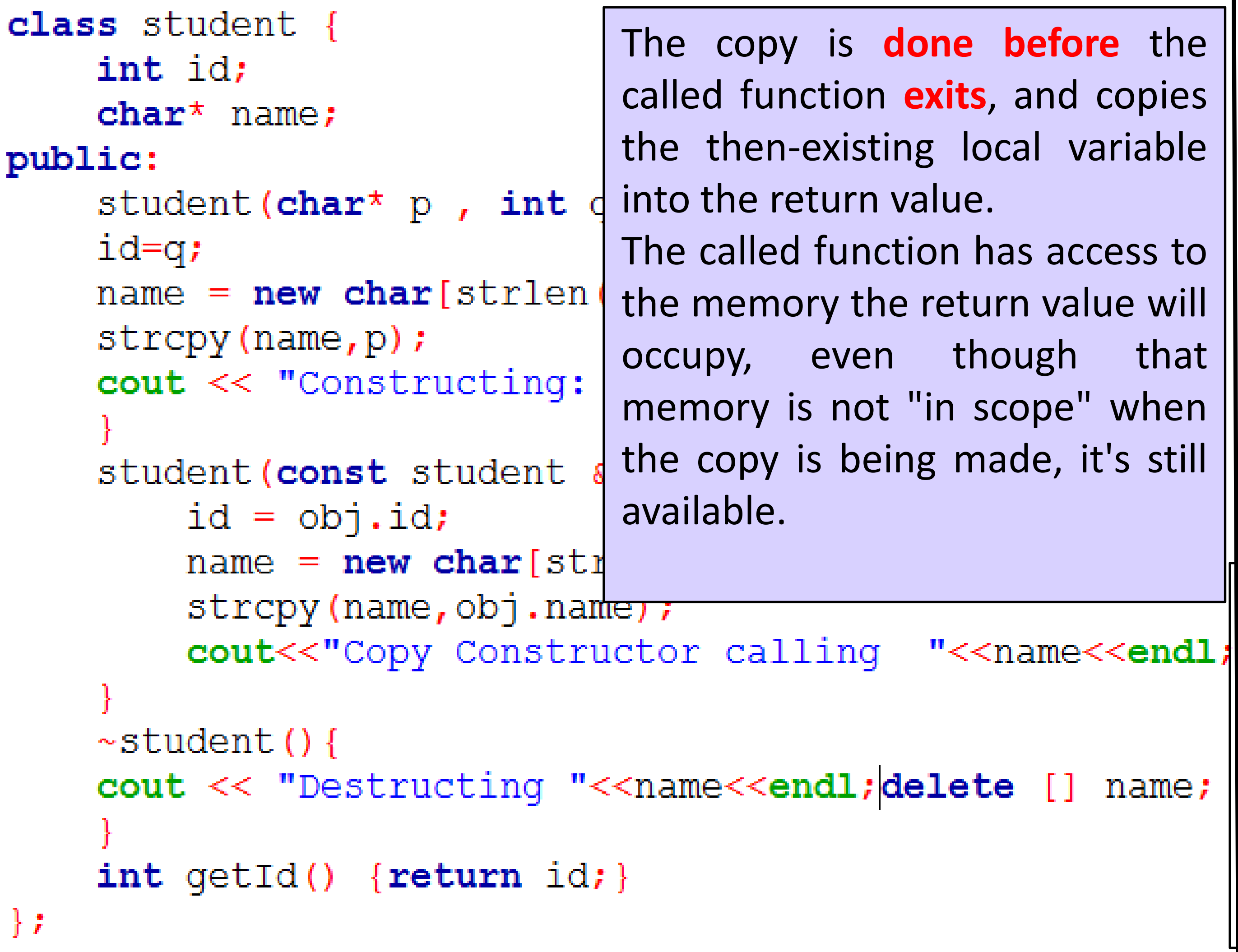
```
class student {
    int id;
    char* name;
public:
    student(char* p , int q) {
    id=q;
    name = new char[strlen(p)];
    strcpy(name,p);
    cout << "Constructing: "<<name<<endl;
    }
    student(const student &obj){
        id = obj.id;
        name = new char[strlen(obj.name)+1];
        strcpy(name,obj.name);
        cout<<"Copy Constructor calling  "<<name<<endl;
    }
    ~student(){
    cout << "Destructing "<<name<<endl;delete [] name;
    }
    int getId() {return id;}
};
```



```
student func(student st){
student temp("temp",3);
return temp;
}

int main(){
student st1("St1",1), st2("st2",2);
student st3 = st1;
st3 = func(st1);
cout<<"Finish"<<endl;
}
```

1.Constructing: St1(In main)
2.Constructing: St2(In main)
3.Copy Constructor Calling St1 (In main)
4.Copy Constructor Calling St1 (In function parameter)
6.Constructing: temp(In local)
7.Copy Constructor Calling temp(in local)

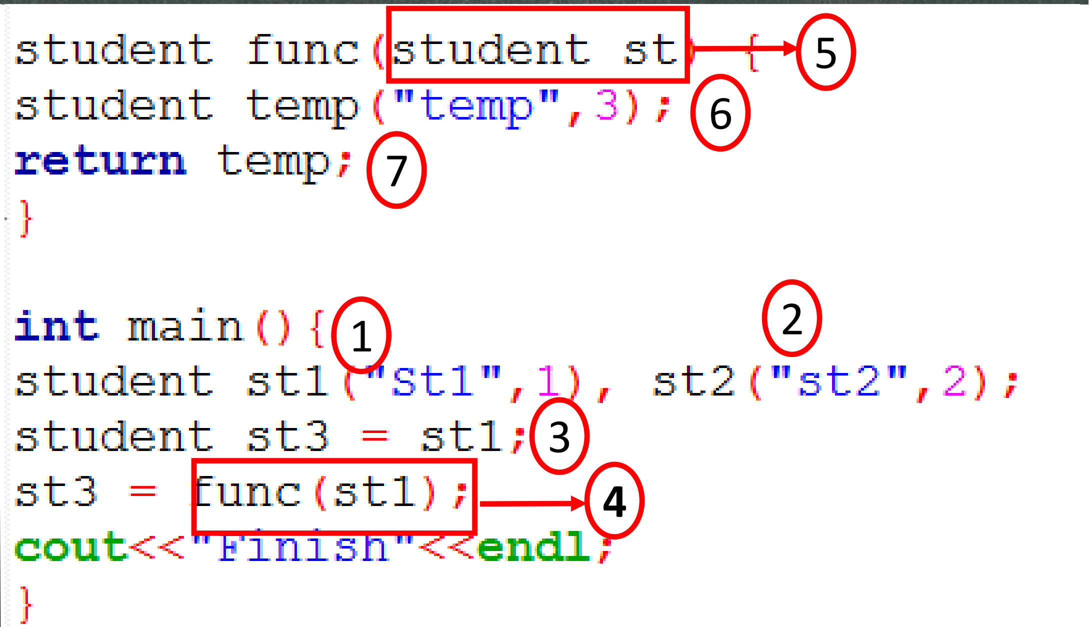
Copy Constructor Invocation Example



```
class student {
    int id;
    char* name;
public:
    student(char* p , int q
    id=q;
    name = new char[strlen(
    strcpy(name,p);
    cout << "Constructing:
    }
    student(const student &
        id = obj.id;
        name = new char[str
        strcpy(name,obj.name);
        cout<<"Copy Constructor calling  "<<name<<endl;
    }
    ~student(){
    cout << "Destructing "<<name<<endl;delete [] name;
    }
    int getId() {return id;}
};
```

The copy is **done before** the called function **exits**, and copies the then-existing local variable into the return value.

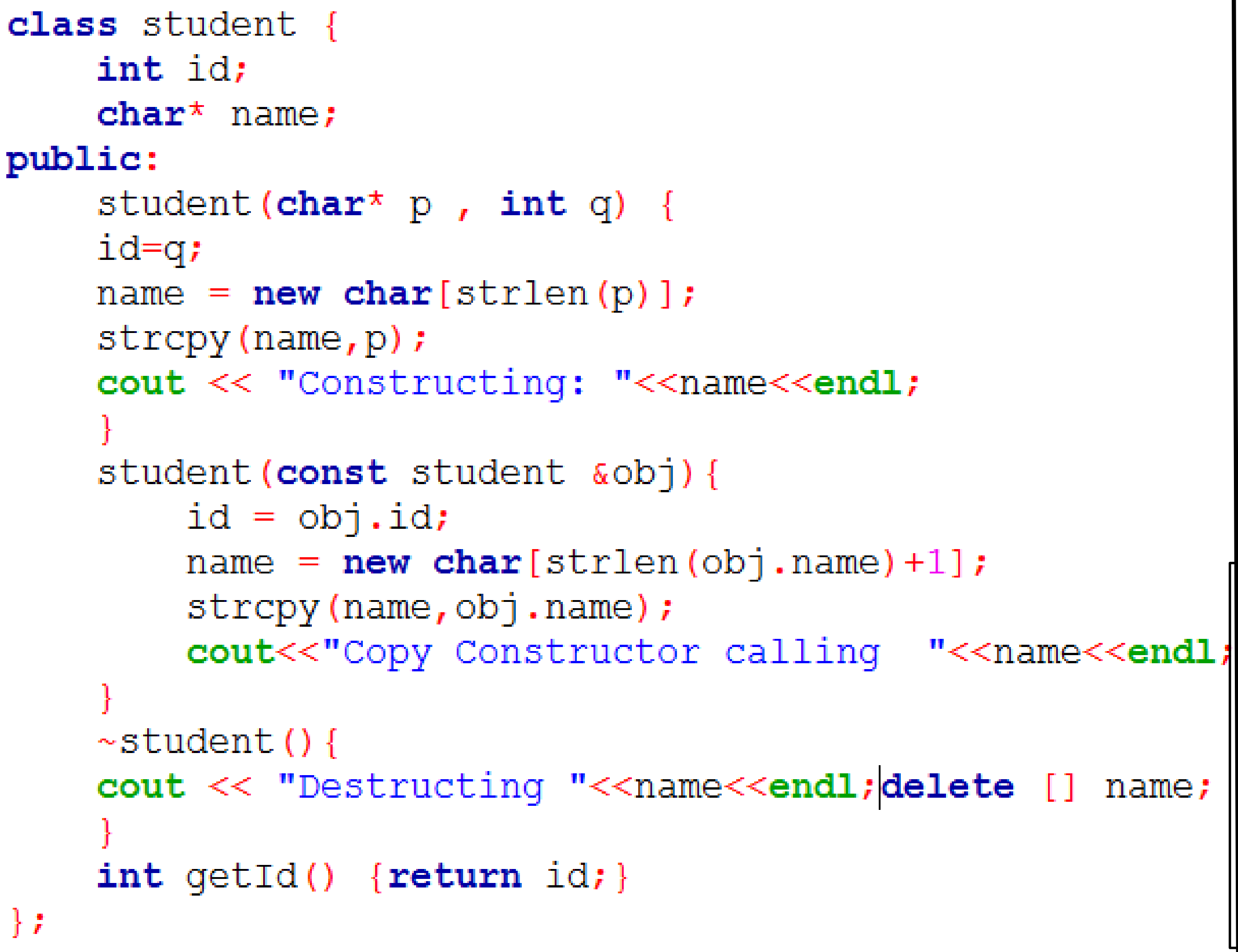
The called function has access to the memory the return value will occupy, even though that memory is not "in scope" when the copy is being made, it's still available.



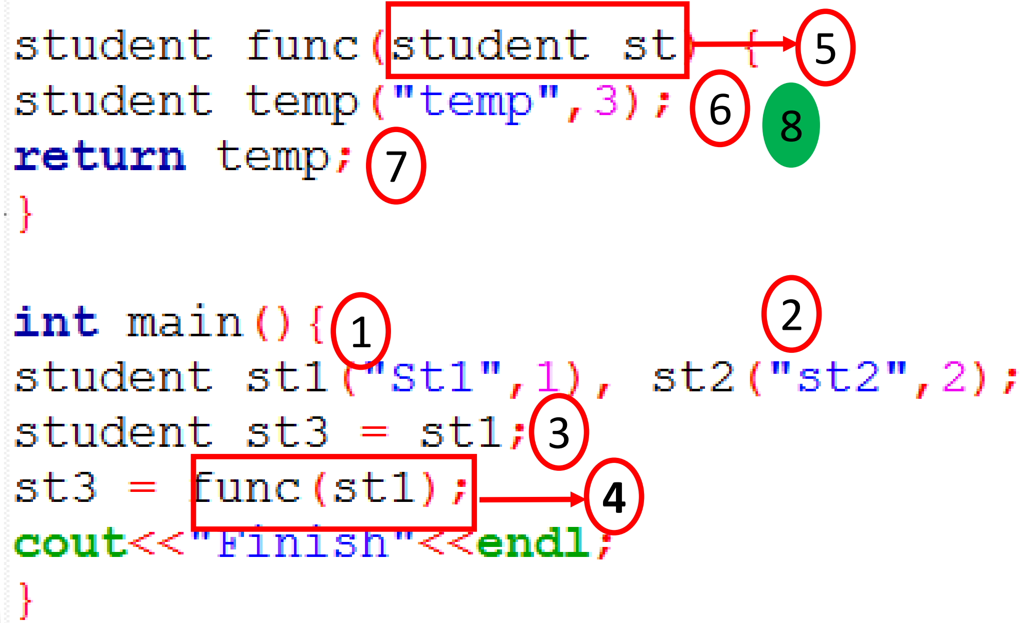
```
student func(student st) {
student temp("temp",3);
return temp;
}

int main(){
student st1("St1",1), st2("st2",2);
student st3 = st1;
st3 = func(st1);
cout<<"Finish"<<endl;
}
```

1.Constructing: St1(In main)
2.Constructing: St2(In main)
3.Copy Constructor Calling St1 (In main)
4.Copy Constructor Calling St1 (In function parameter)
6.Constructing: temp(In local)
7.Copy Constructor Calling temp(in local)

Copy Constructor Invocation Example



```
class student {
    int id;
    char* name;
public:
    student(char* p , int q) {
    id=q;
    name = new char[strlen(p)];
    strcpy(name,p);
    cout << "Constructing: "<<name<<endl;
    }
    student(const student &obj){
        id = obj.id;
        name = new char[strlen(obj.name)+1];
        strcpy(name,obj.name);
        cout<<"Copy Constructor calling  "<<name<<endl;
    }
    ~student(){
    cout << "Destructing "<<name<<endl;delete [] name;
    }
    int getId() {return id;}
};
```

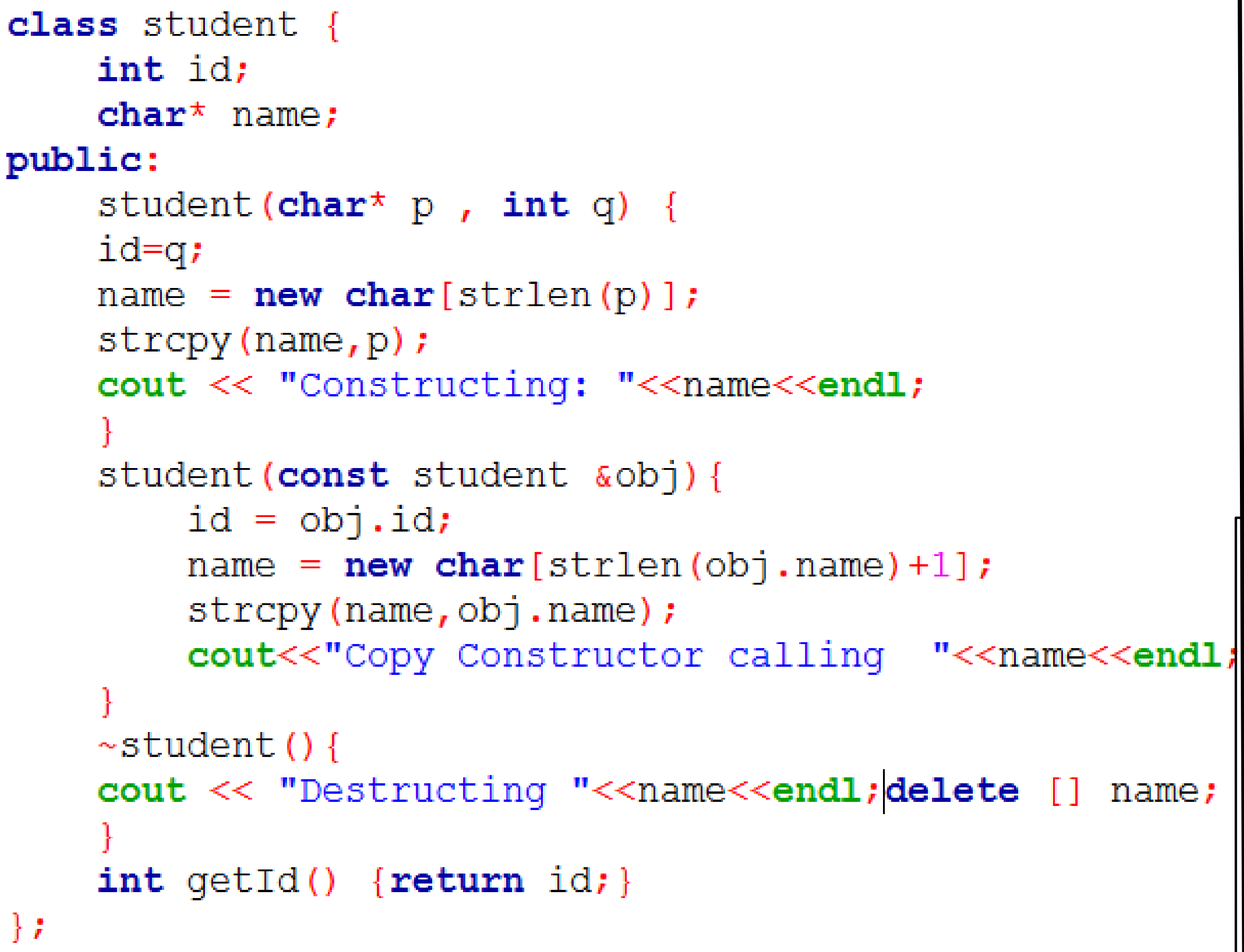


```
student func(student st){
student temp("temp",3);
return temp;
}

int main(){
student st1("St1",1), st2("st2",2);
student st3 = st1;
st3 = func(st1);
cout<<"Finish"<<endl;
}
```

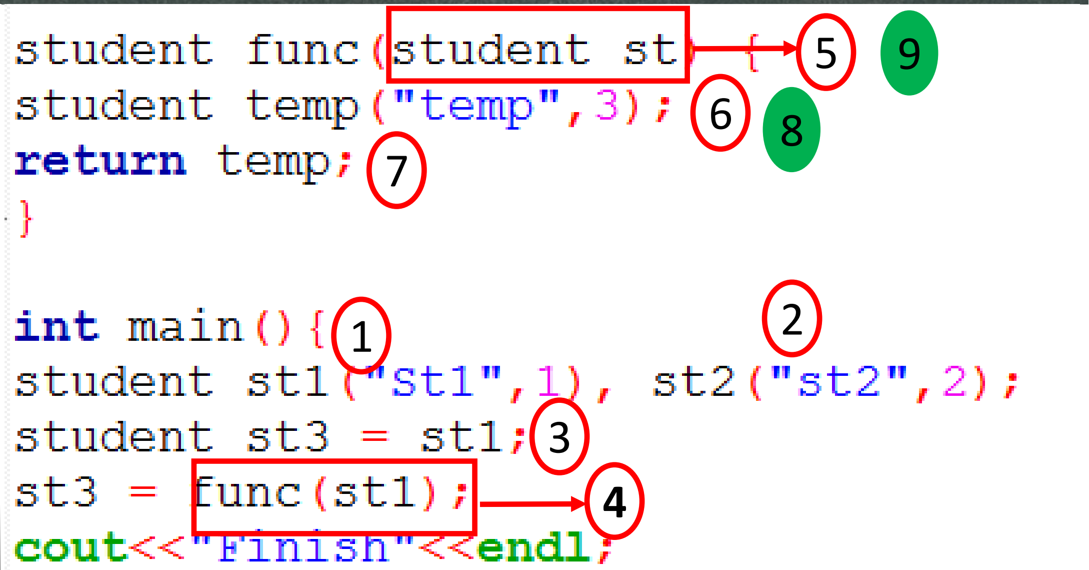
1.Constructing: St1(In main)
2.Constructing: St2(In main)
3.Copy Constructor Calling St1 (In main)
4.Copy Constructor Calling St1 (In function parameter)
6.Constructing: temp(In local)
7.Copy Constructor Calling temp(in local)

8.Destructing temp(local)

Copy Constructor Invocation Example



```
class student {
    int id;
    char* name;
public:
    student(char* p , int q) {
    id=q;
    name = new char[strlen(p)];
    strcpy(name,p);
    cout << "Constructing: "<<name<<endl;
    }
    student(const student &obj){
        id = obj.id;
        name = new char[strlen(obj.name)+1];
        strcpy(name,obj.name);
        cout<<"Copy Constructor calling  "<<name<<endl;
    }
    ~student(){
    cout << "Destructing "<<name<<endl;delete [] name;
    }
    int getId() {return id;}
};
```

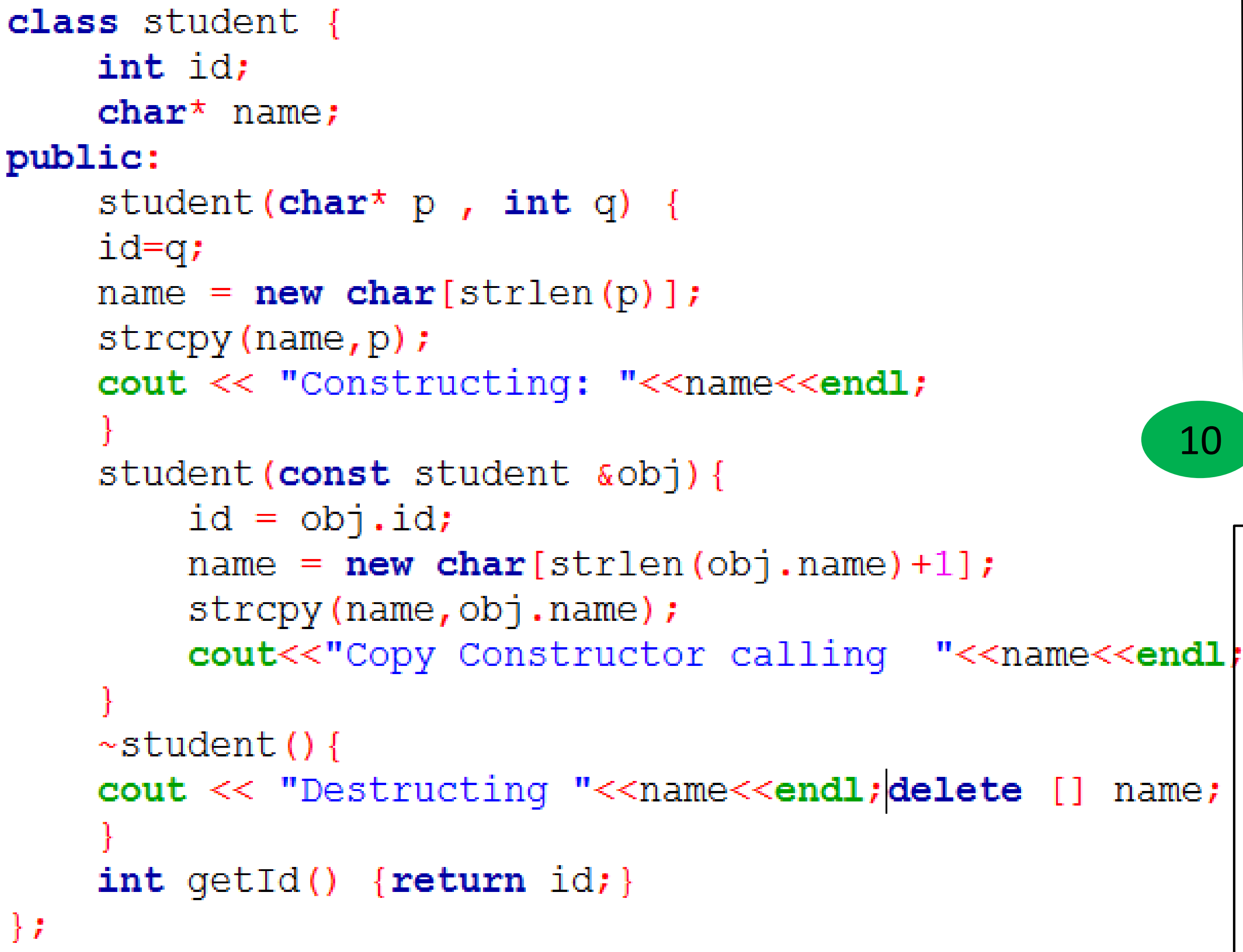


```
student func(student st){
student temp("temp",3);
return temp;
}

int main(){
student st1("St1",1), st2("st2",2);
student st3 = st1;
st3 = func(st1);
cout<<"Finish"<<endl;
}
```

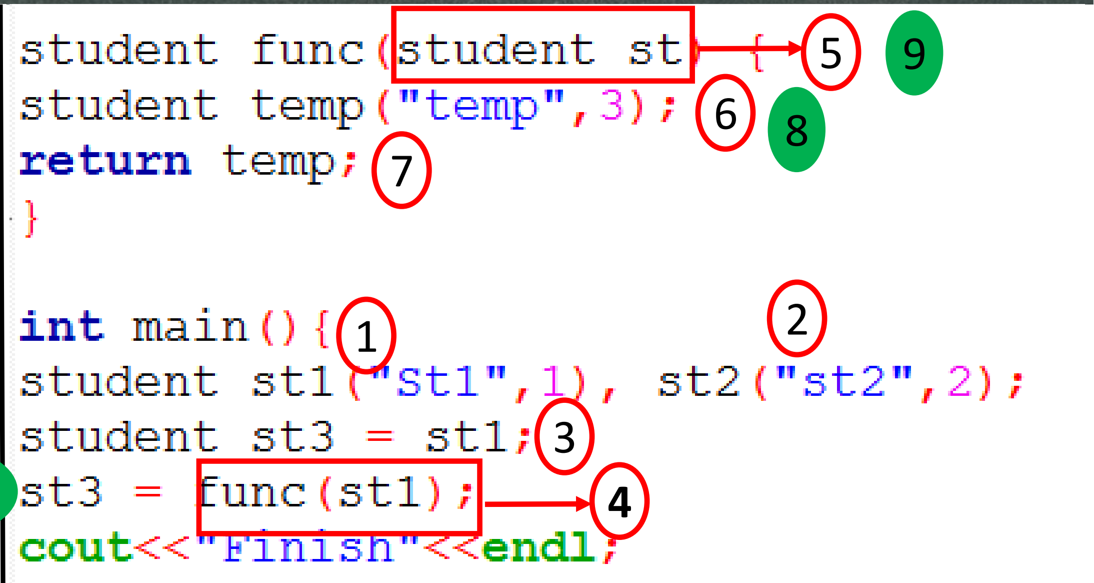
1.Constructing: St1(In main)
2.Constructing: St2(In main)
3.Copy Constructor Calling St1 (In main)
4.Copy Constructor Calling St1 (In function parameter)
6.Constructing: temp(In local)
7.Copy Constructor Calling temp(in local)

8.Destructing temp(local)
9.Destructing St1(local)

Copy Constructor Invocation Example



```
class student {
    int id;
    char* name;
public:
    student(char* p , int q) {
    id=q;
    name = new char[strlen(p)];
    strcpy(name,p);
    cout << "Constructing: "<<name<<endl;
    }
    student(const student &obj){
        id = obj.id;
        name = new char[strlen(obj.name)+1];
        strcpy(name,obj.name);
        cout<<"Copy Constructor calling  "<<name<<endl;
    }
    ~student(){
    cout << "Destructing "<<name<<endl;delete [] name;
    }
    int getId() {return id;}
};
```

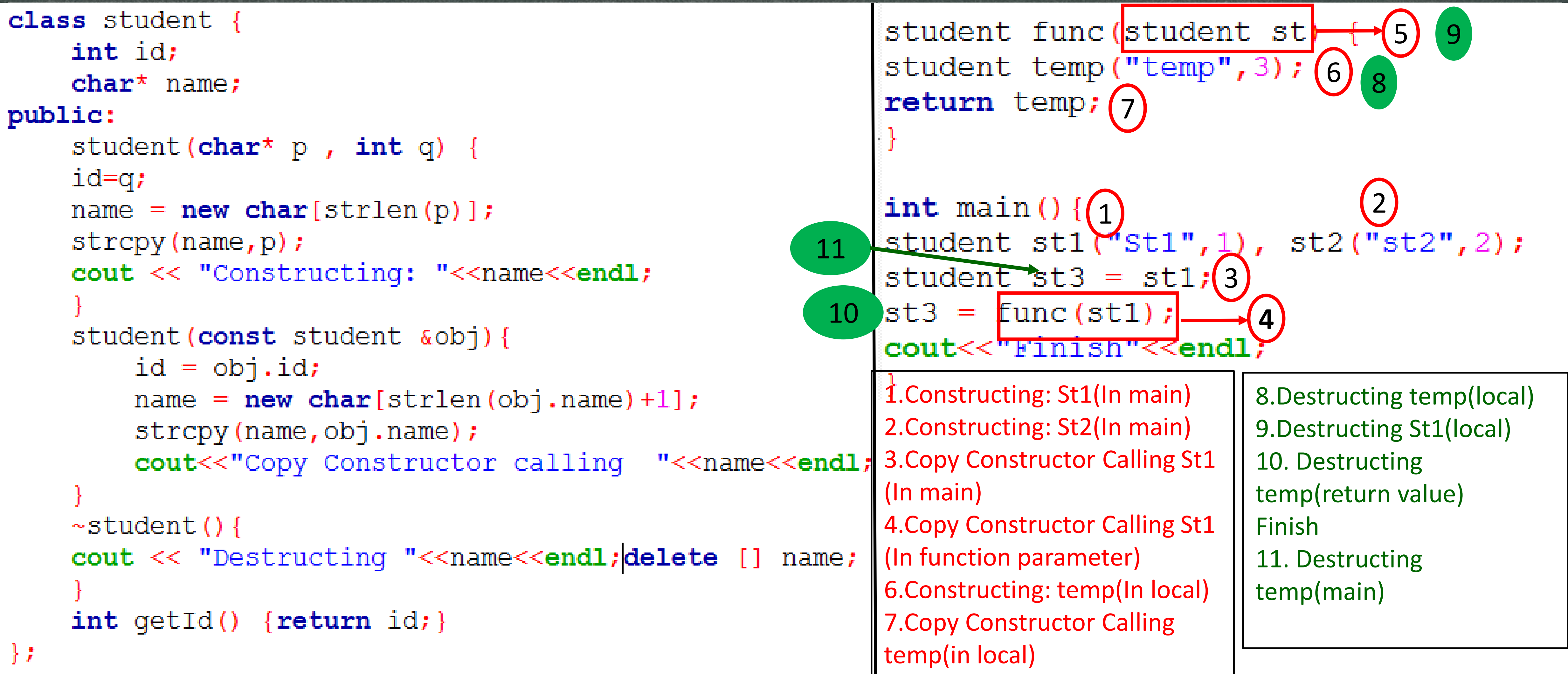


```
student func(student st){
student temp("temp",3);
return temp;
}

int main(){
student st1("St1",1), st2("st2",2);
student st3 = st1;
st3 = func(st1);
cout<<"Finish"<<endl;
}
```

1.Constructing: St1(In main)
2.Constructing: St2(In main)
3.Copy Constructor Calling St1 (In main)
4.Copy Constructor Calling St1 (In function parameter)
6.Constructing: temp(In local)
7.Copy Constructor Calling temp(in local)

8.Destructing temp(local)
9.Destructing St1(local)
10. Destructing temp(return value)

Copy Constructor Invocation Example



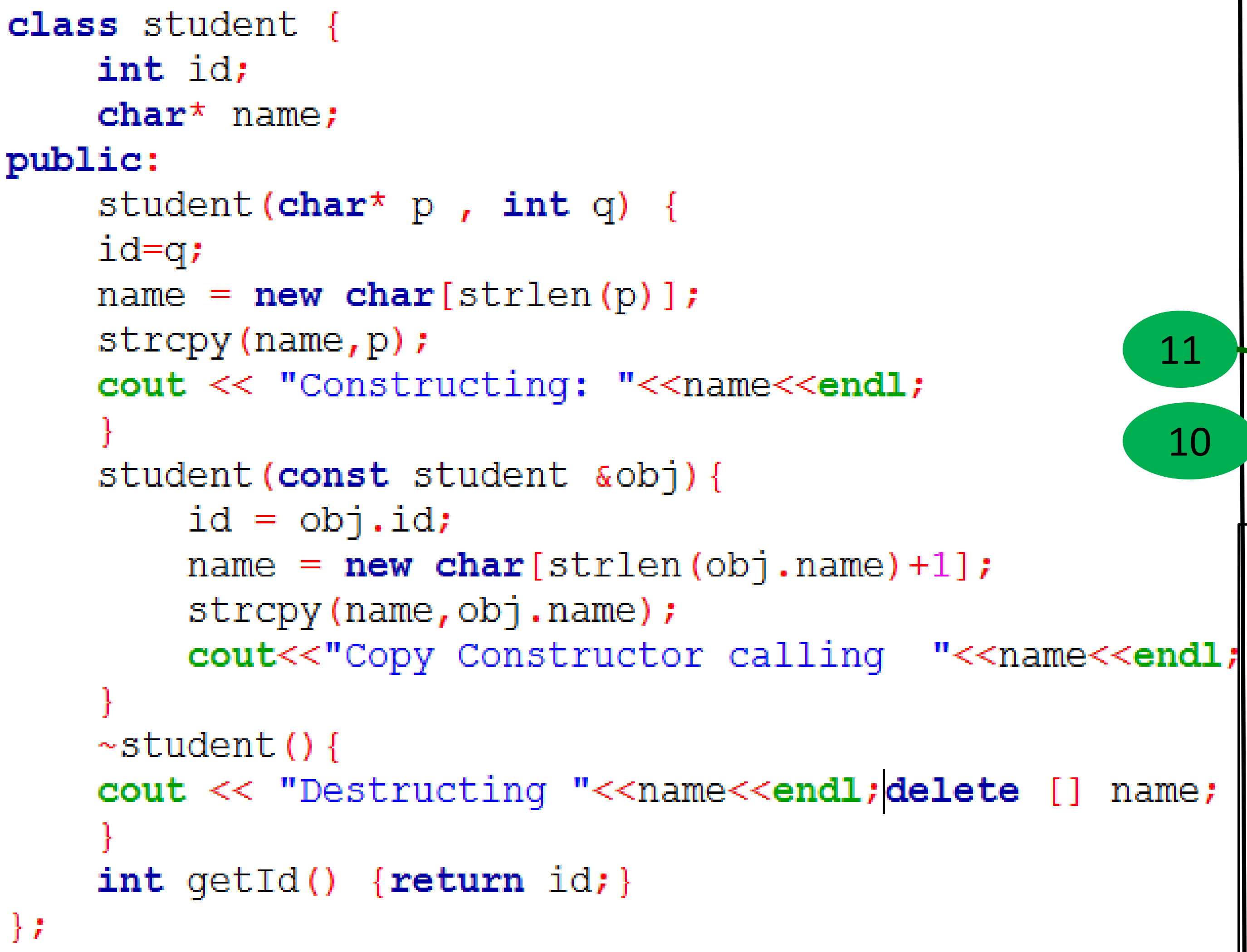
```
class student {
    int id;
    char* name;
public:
    student(char* p , int q) {
    id=q;
    name = new char[strlen(p)];
    strcpy(name,p);
    cout << "Constructing: "<<name<<endl;
    }
    student(const student &obj){
        id = obj.id;
        name = new char[strlen(obj.name)+1];
        strcpy(name,obj.name);
        cout<<"Copy Constructor calling  "<<name<<endl;
    }
    ~student(){
    cout << "Destructing "<<name<<endl;delete [] name;
    }
    int getId() {return id;}
};
```

```
student func(student st){
student temp("temp",3);
return temp;
}

int main(){
student st1("St1",1), st2("st2",2);
student st3 = st1;
st3 = func(st1);
cout<<"Finish"<<endl;
}
```

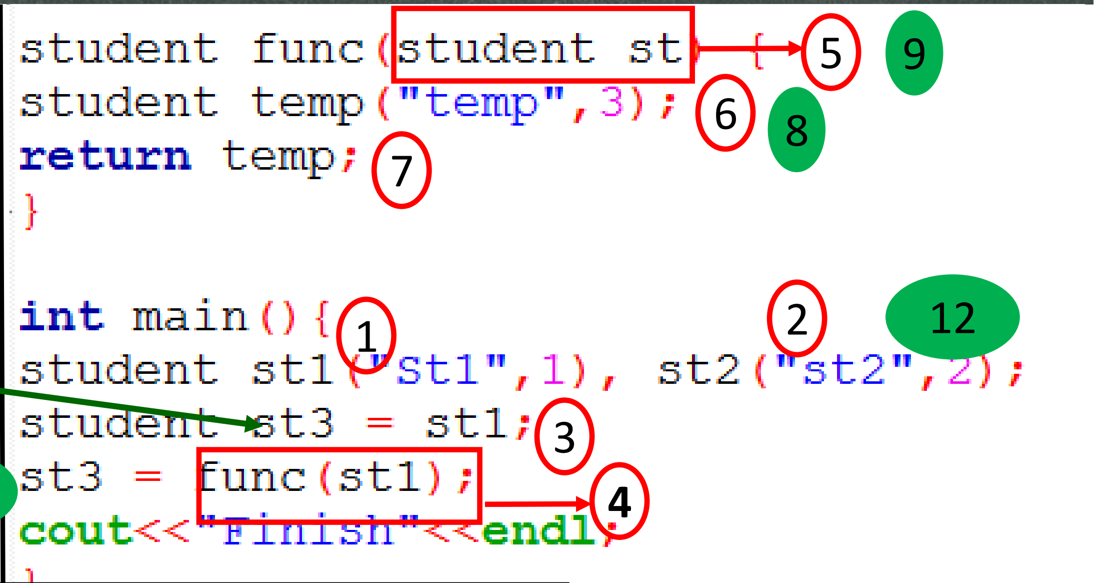
1.Constructing: St1(In main)
2.Constructing: St2(In main)
3.Copy Constructor Calling St1 (In main)
4.Copy Constructor Calling St1 (In function parameter)
6.Constructing: temp(In local)
7.Copy Constructor Calling temp(in local)

8.Destructing temp(local)
9.Destructing St1(local)
10. Destructing temp(return value)
Finish
11. Destructing temp(main)

Copy Constructor Invocation Example



```
class student {
    int id;
    char* name;
public:
    student(char* p , int q) {
    id=q;
    name = new char[strlen(p)];
    strcpy(name,p);
    cout << "Constructing: "<<name<<endl;
    }
    student(const student &obj){
        id = obj.id;
        name = new char[strlen(obj.name)+1];
        strcpy(name,obj.name);
        cout<<"Copy Constructor calling  "<<name<<endl;
    }
    ~student(){
    cout << "Destructing "<<name<<endl;delete [] name;
    }
    int getId() {return id;}
};
```

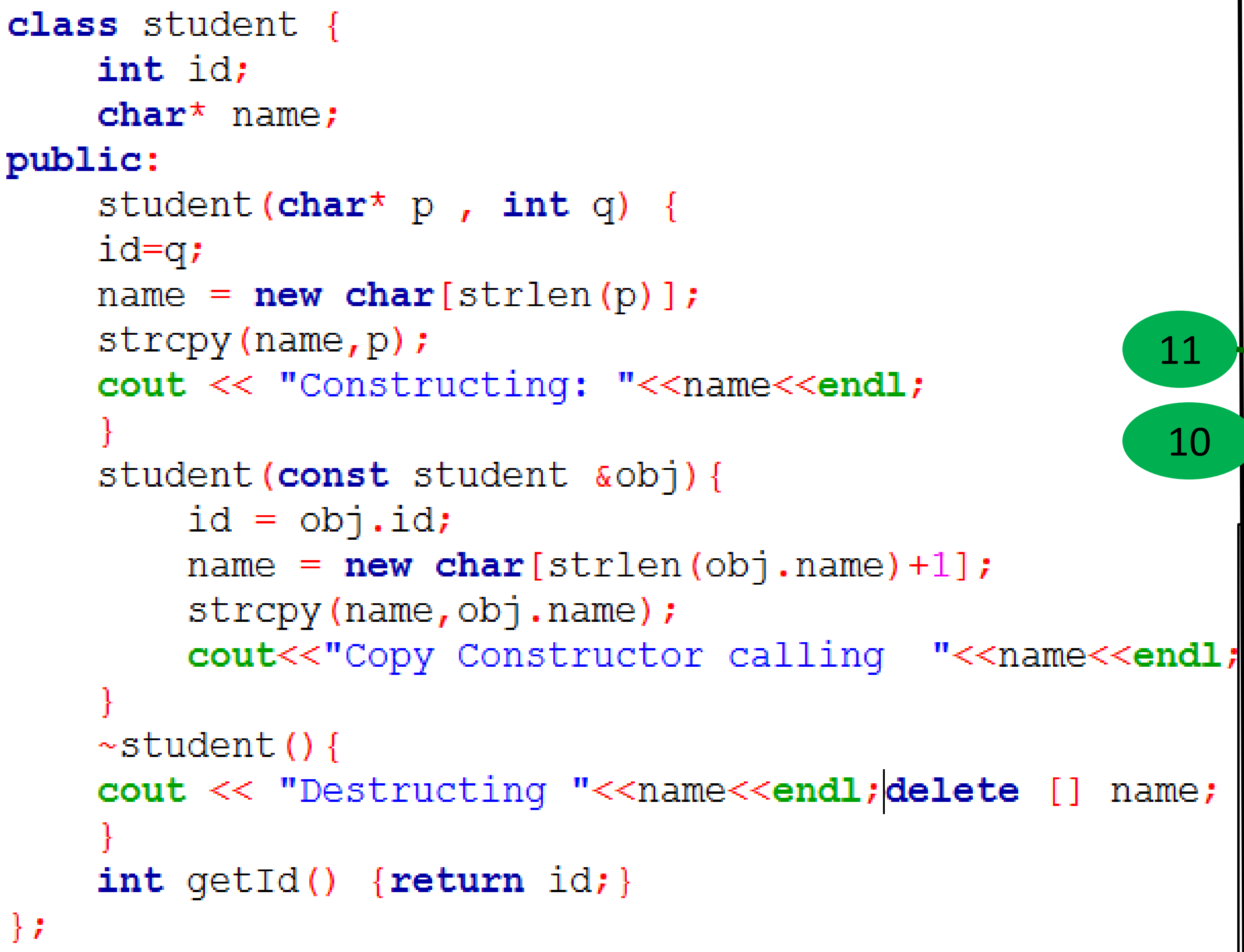


```
student func(student st){
student temp("temp",3);
return temp;
}

int main(){
student st1("St1",1), st2("st2",2);
student st3 = st1;
st3 = func(st1);
cout<<"Finish"<<endl;
}
```

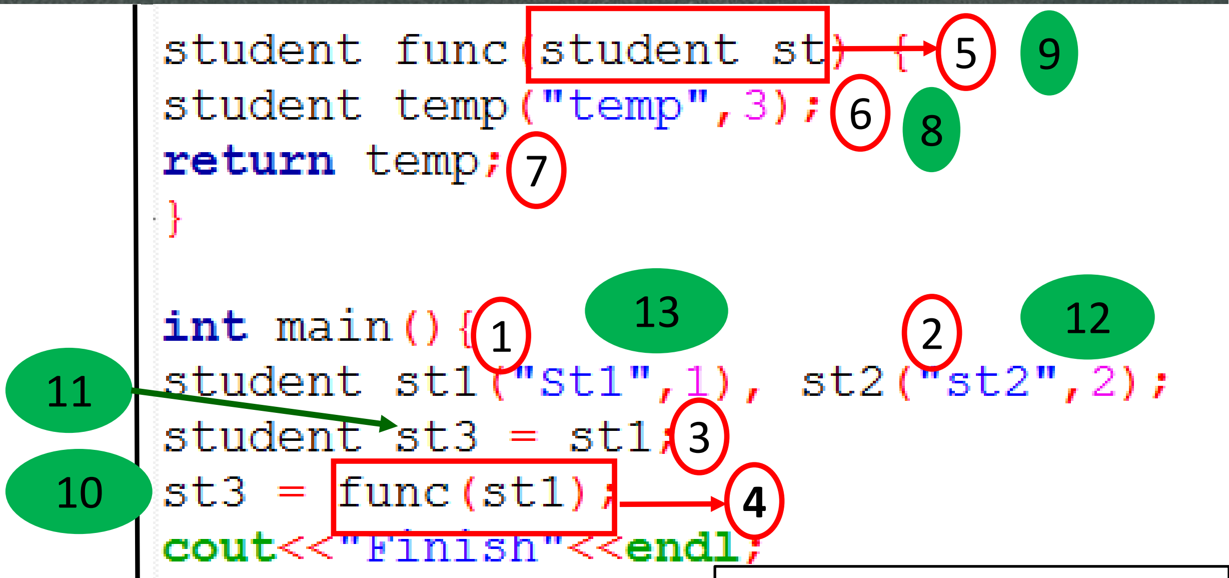
1.Constructing: St1(In main)
2.Constructing: St2(In main)
3.Copy Constructor Calling St1 (In main)
4.Copy Constructor Calling St1 (In function parameter)
6.Constructing: temp(In local)
7.Copy Constructor Calling temp(in local)

8.Destructing temp(local)
9.Destructing St1(local)
10. Destructing temp(return value)
Finish
11. Destructing temp(main)
12. Destructing St2(main)

Copy Constructor Invocation Example



```
class student {
    int id;
    char* name;
public:
    student(char* p , int q) {
    id=q;
    name = new char[strlen(p)];
    strcpy(name,p);
    cout << "Constructing: "<<name<<endl;
    }
    student(const student &obj){
        id = obj.id;
        name = new char[strlen(obj.name)+1];
        strcpy(name,obj.name);
        cout<<"Copy Constructor calling  "<<name<<endl;
    }
    ~student(){
    cout << "Destructing "<<name<<endl;delete [] name;
    }
    int getId() {return id;}
};
```

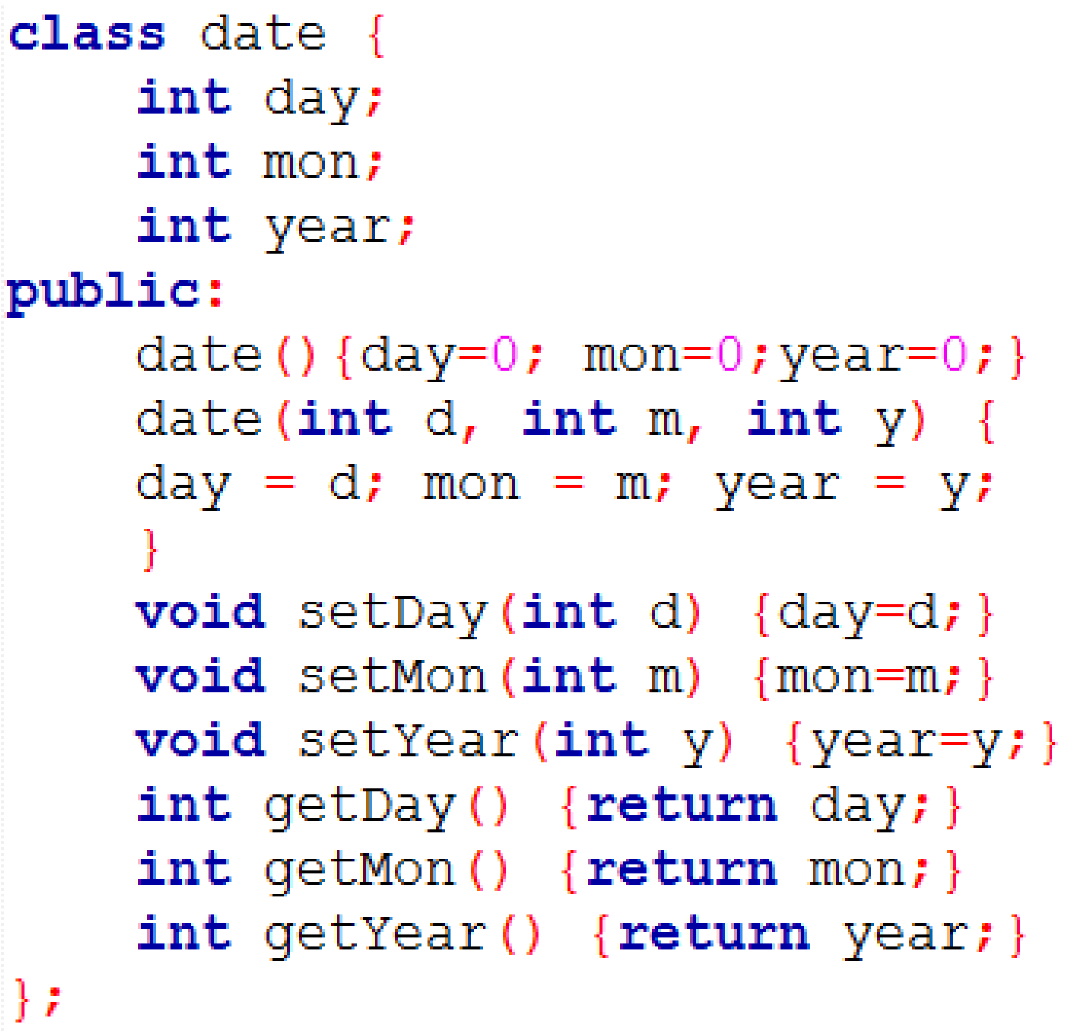


```
student func(student st) {
student temp("temp",3);
return temp;
}

int main() {
student st1("St1",1), st2("st2",2);
student st3 = st1;
st3 = func(st1);
cout<<"Finish"<<endl;
}
```

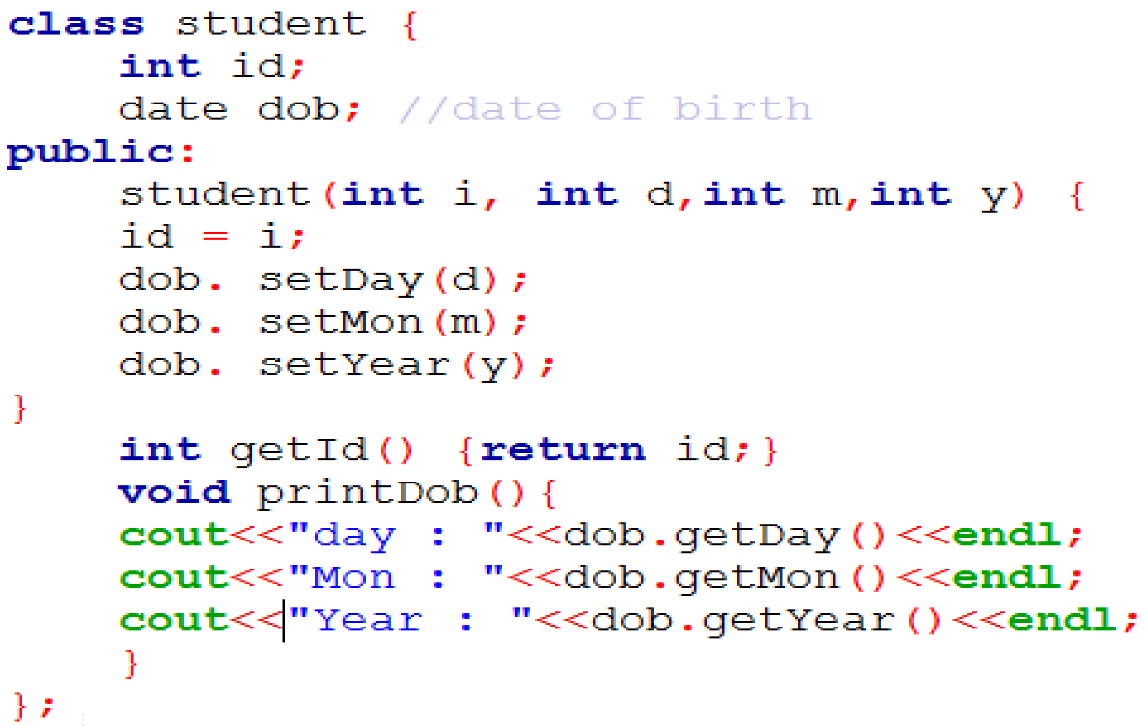
1.Constructing: St1(In main)
2.Constructing: St2(In main)
3.Copy Constructor Calling St1 (In main)
4.Copy Constructor Calling St1 (In function parameter)
6.Constructing: temp(In local)
7.Copy Constructor Calling temp(in local)

8.Destructing temp(local)
9.Destructing St1(local)
10. Destructing temp(return value)
Finish
11 Destructing temp(main)
12. Destructing St2(main)
13. Destructing St1(main)

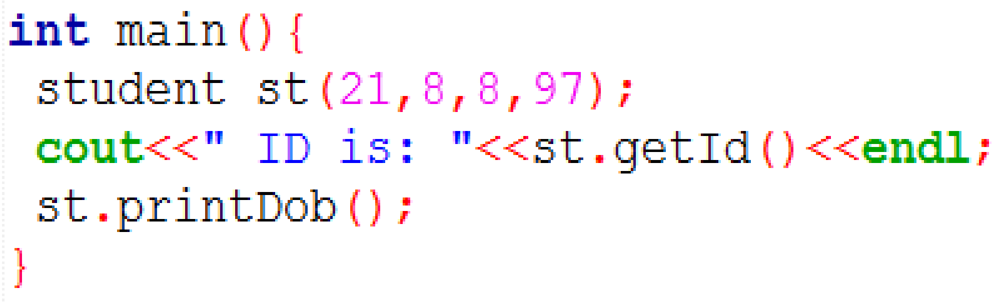
Class Member of Another Class



```
class date {
    int day;
    int mon;
    int year;
public:
    date(){day=0; mon=0;year=0;}
    date(int d, int m, int y) {
    day = d; mon = m; year = y;
    }
    void setDay(int d) {day=d;}
    void setMon(int m) {mon=m;}
    void setYear(int y) {year=y;}
    int getDay() {return day;}
    int getMon() {return mon;}
    int getYear() {return year;}
};
```



```
class student {
     int id;
     date dob; //date of birth
public:
     student(int i, int d,int m,int y) {
     id = i;
     dob. setDay(d);
     dob. setMon(m);
     dob. setYear(y);
}
     int getId() {return id;}
     void printDob(){
     cout<<"day : "<<dob.getDay()<<endl;
     cout<<"Mon : "<<dob.getMon()<<endl;
     cout<<"Year : "<<dob.getYear()<<endl;
     }
};
```



```
int main(){
 student st(21,8,8,97);
 cout<<" ID is: "<<st.getId()<<endl;
 st.printDob();
}
```

Thank you!