Pointers - intro

```
int x;
int * p;
…
```

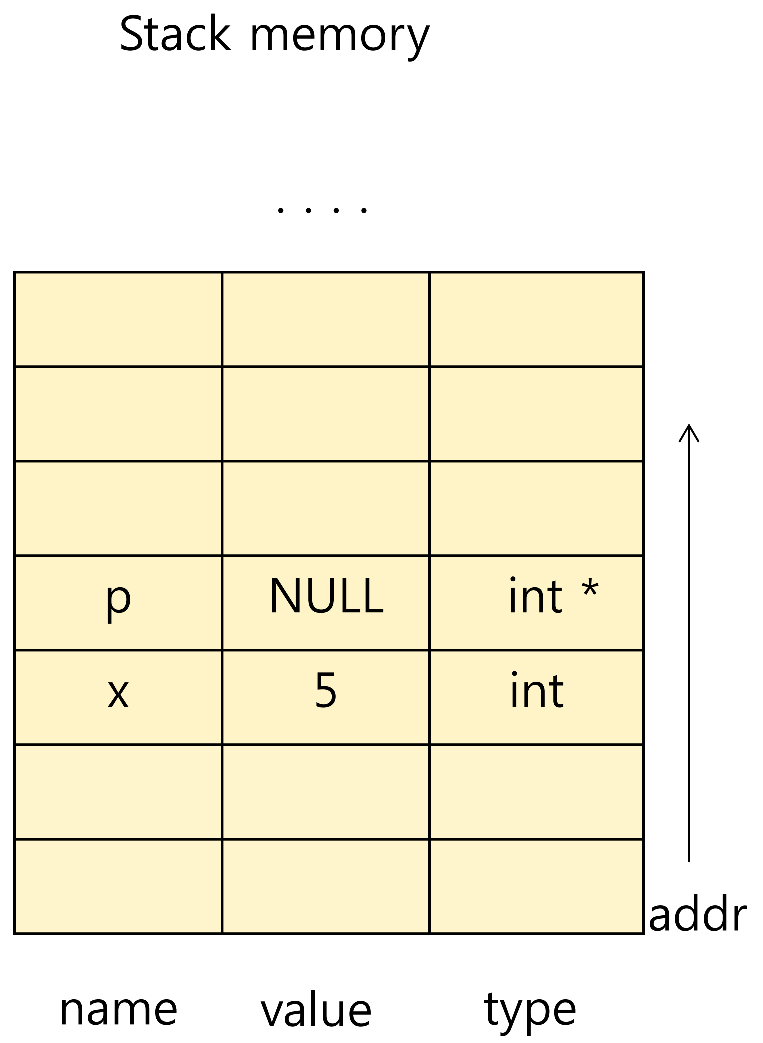
How do we assign to p?

```
P =

P =
```

______________ `operator: &`

______________ `operator: *`



Exercise 1

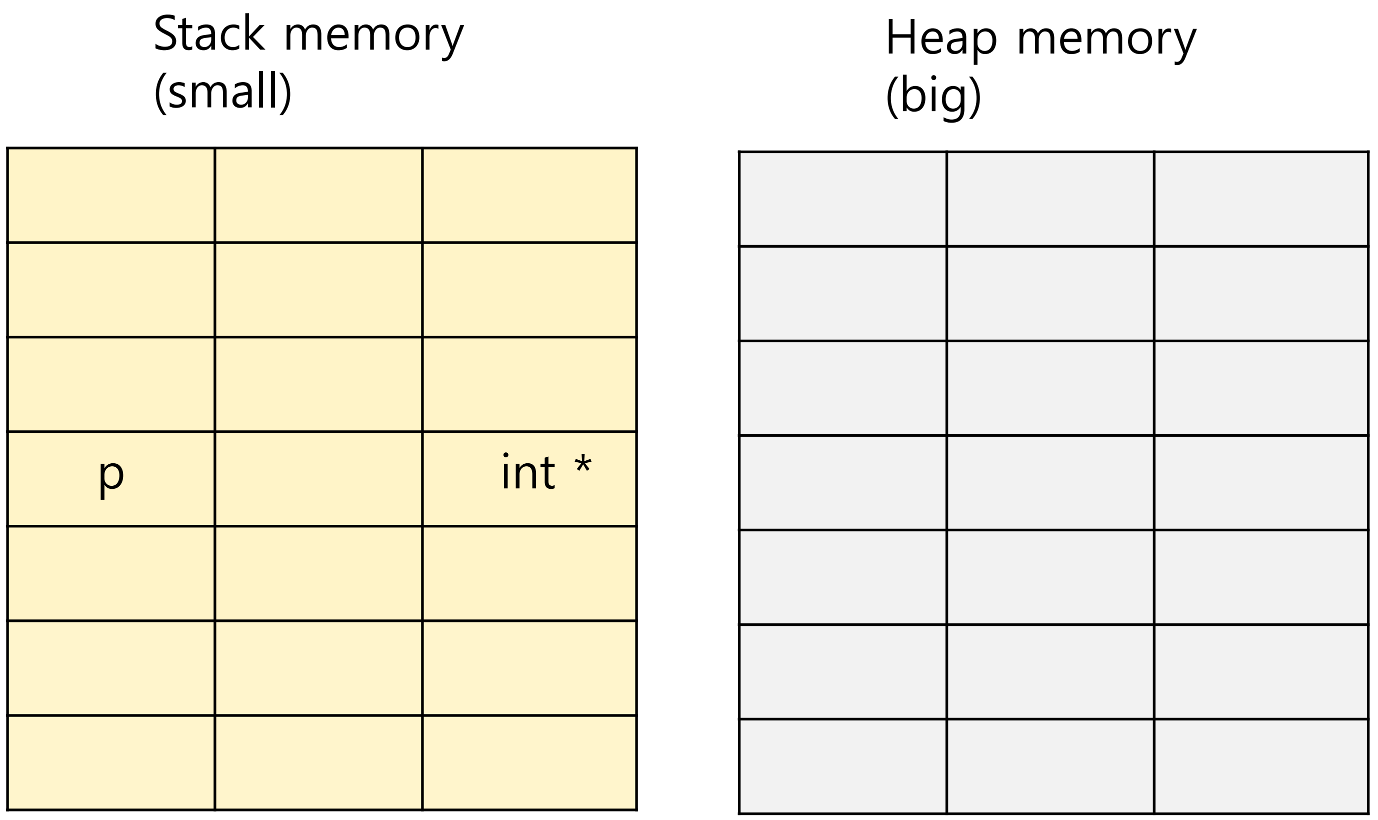
```
#include "sphere.h"
using namespace std;

Sphere *CreateUnitSphere() {
   Sphere s(1);
   return &s;
}

int main() {
   Sphere *s =
CreateUnitSphere();
   double r = s->getDiameter();
   double v = s->getAtra();
   return 0;
}
```

Pointer variables and dynamic memory allocation

```
int * p;
```



Exercise 2

```
int * p, q;
```
What type is q? __________

```
int *p;
int x;
p = &x;
*p = 6;
cout << x;
```
What is output? ________

```
cout << p;
```
What is output? ________

Write a statement whose output is the value of `x`, using variable `p`:

____

```
int *p, *q;

p = new int;

q = p;

*q = 8;

cout << *p;         What is output? _______

q = new int;

*q = 9;

p = NULL;           // __________

delete q;

q = NULL;           //  __________
```

Memory leak:
Deleting a null pointer:
Dereferencing a null pointer:

```
int *p, *q;

p = new int;

q = p;

delete p;

…

cout << *q;    // _____________
```

Stack vs. Heap memory:

```
void func() {
    string s = “hello!”;
    cout << s << endl;
}

int main() {
   func();
   return 0;
}
```

```
void func() {
    string *s = new string;
    *s = “hello?”;
    cout << s << endl;
    delete s;
}

int main() {
   func();
   return 0;
}
```

System allocates space for s and takes care of freeing it when s goes out of scope.

Data can be accesses directly, rather than via a pointer.

Allocated memory must be deleted

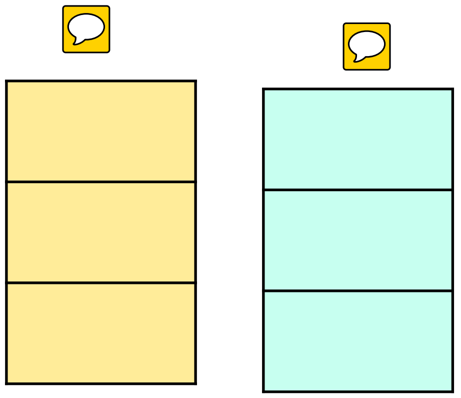
____

Data ________ be accessed by a pointer.

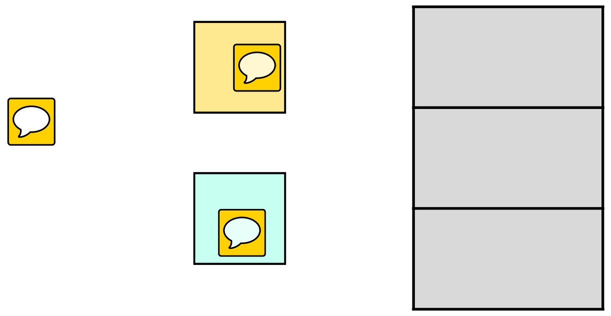
Pointers and objects:

```
class face {
public:
    void setName(string n);
    string getName();
    …
private:
    string name;
    Picture pic;
    boolean done;
};
```

```
face a, b;
… // init b
a = b;
a.setName(“Taewhan”);
b.getName();
```



```
face *c, *d;
… // init *d
c = d;
c->setName(“Byungmin”);
(*d).getName();
```



Exercise 3

```
#include <iostream>
using namespace std;
int main() {
  int *p;
  int x;
  p = &x;
  x = 6;
  cout << x << endl;
  cout << p << endl;
  return 0;
}
```

Exercise 4

```
#include <iostream>
using namespace std;
int main() {
  int *p, *q;
   p = new int;
   q = p;
   *q = 8;
   cout << *p << endl;
   q = new int; *q = 9;
   cout << *p << endl;
   cout << *q << endl;
  return 0;
}
```

Exercise 5

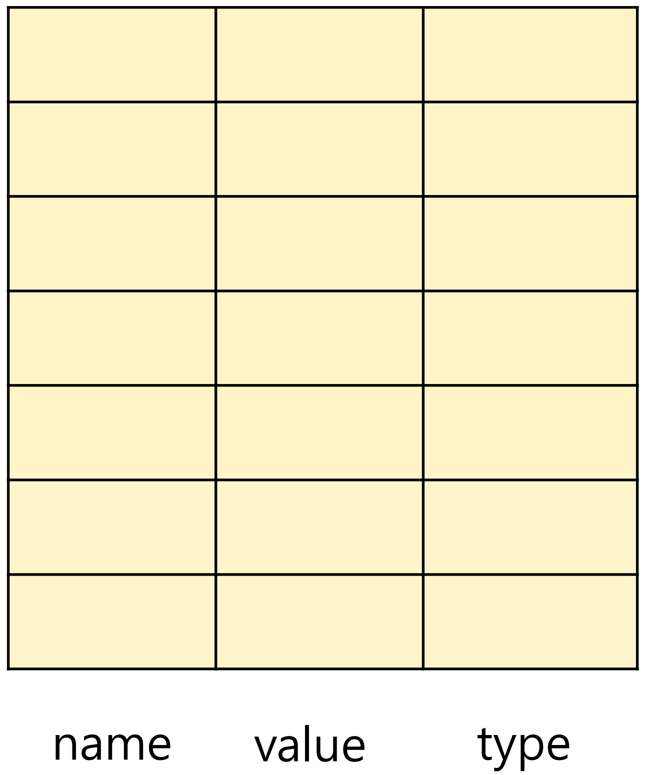
```
#include <iostream>
using namespace std;
int main() {
    Sphere *s1 = new Sphere();
    Sphere *s2 = s1;

    s2->setRadius( 10 );
  return 0;
}
```

Array: static / local (stack)

```
int x[5];
```

Stack memory

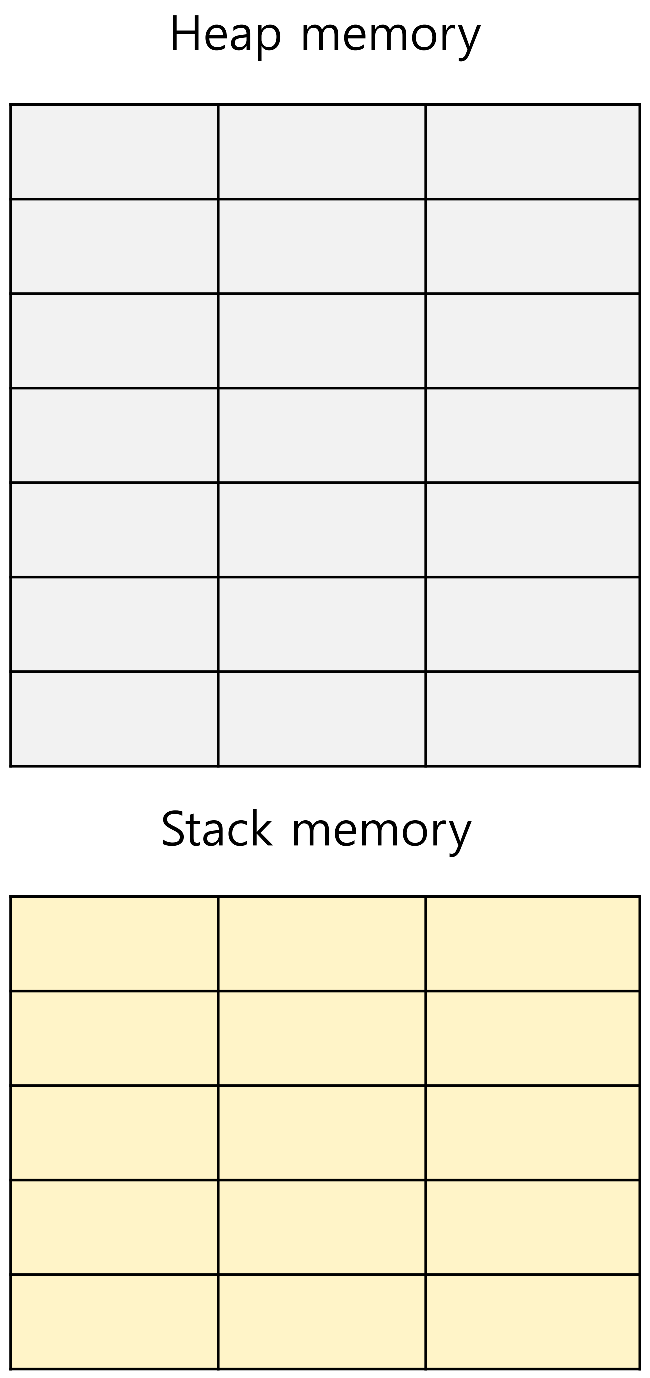


Array: dynamic (heap)

```
int * x;
int size = 3;
x = new int[size];

for (int i=0; i<size; i++)
     x[i] = I + 4;

delete [] x;
```



A point to point: How is my `garden` implemented?

```
class garden {
public:
…
// all the public members
…
private:
  flower ** plot;
  // other stuff
};
```

Option 1:

Option 2:

Option 3:

Option 4: