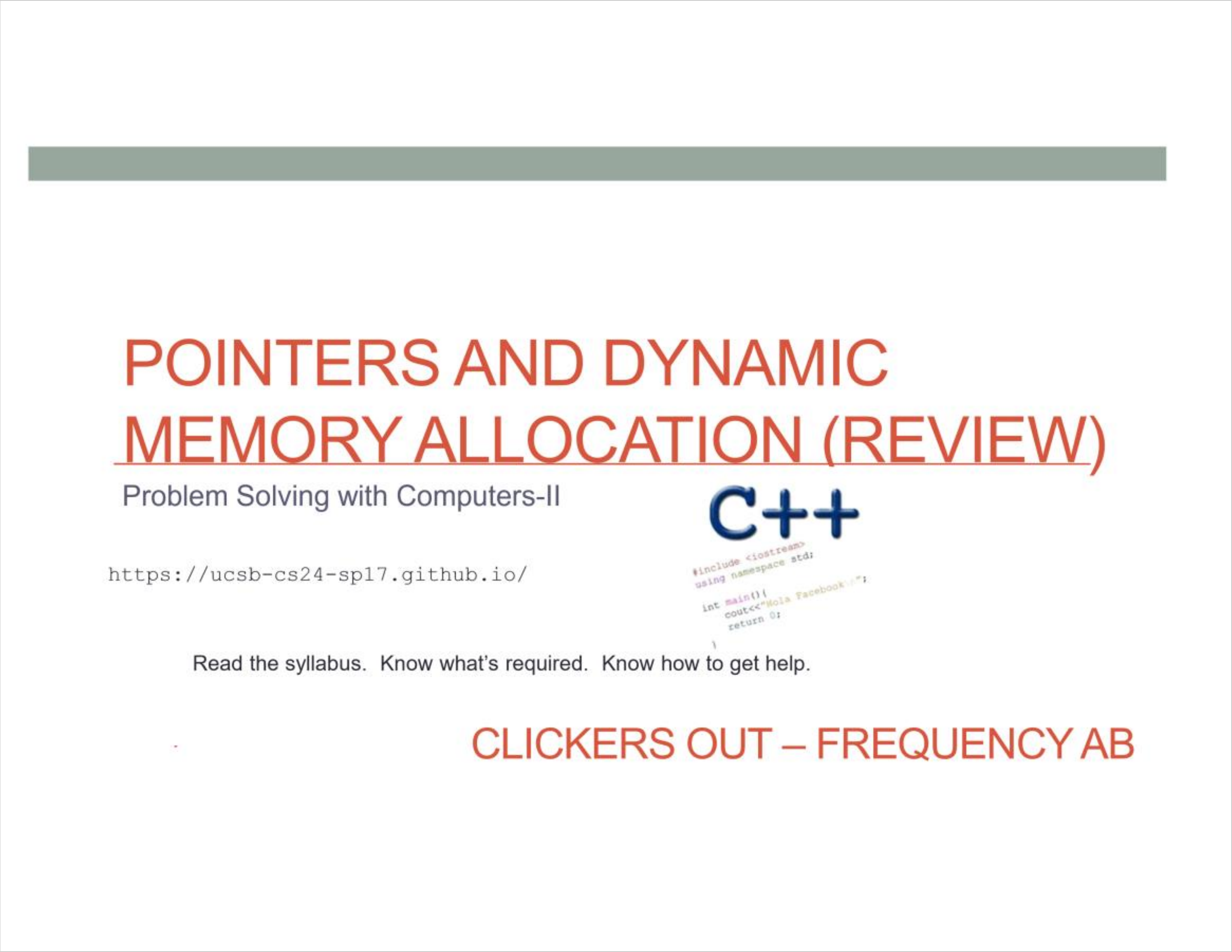
Lecture 7: pointers and dynamic memory allocation

Monday, April 24, 2017 4:34 PM



POINTERS AND DYNAMIC MEMORY ALLOCATION (REVIEW)

Problem Solving with Computers-II

https://ucsb-cs24-sp17.github.io/

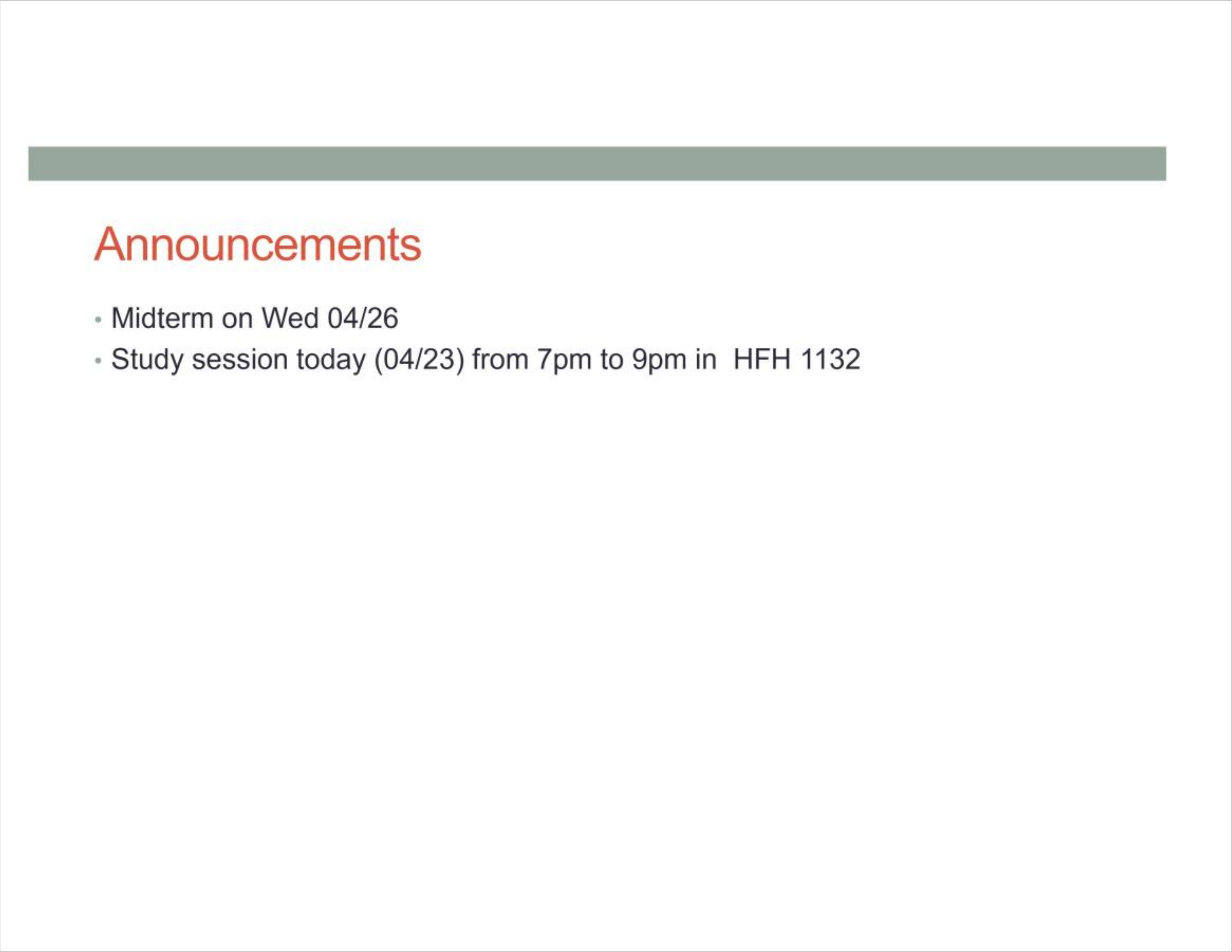
C++
#include <iostream>
using namespace std;
int main(){
cout<<"Hola Facebook :)";
return 0;
}

Read the syllabus. Know what's required. Know how to get help.

CLICKERS OUT – FREQUENCY AB

Page 2

Monday, April 24, 2017 4:34 PM

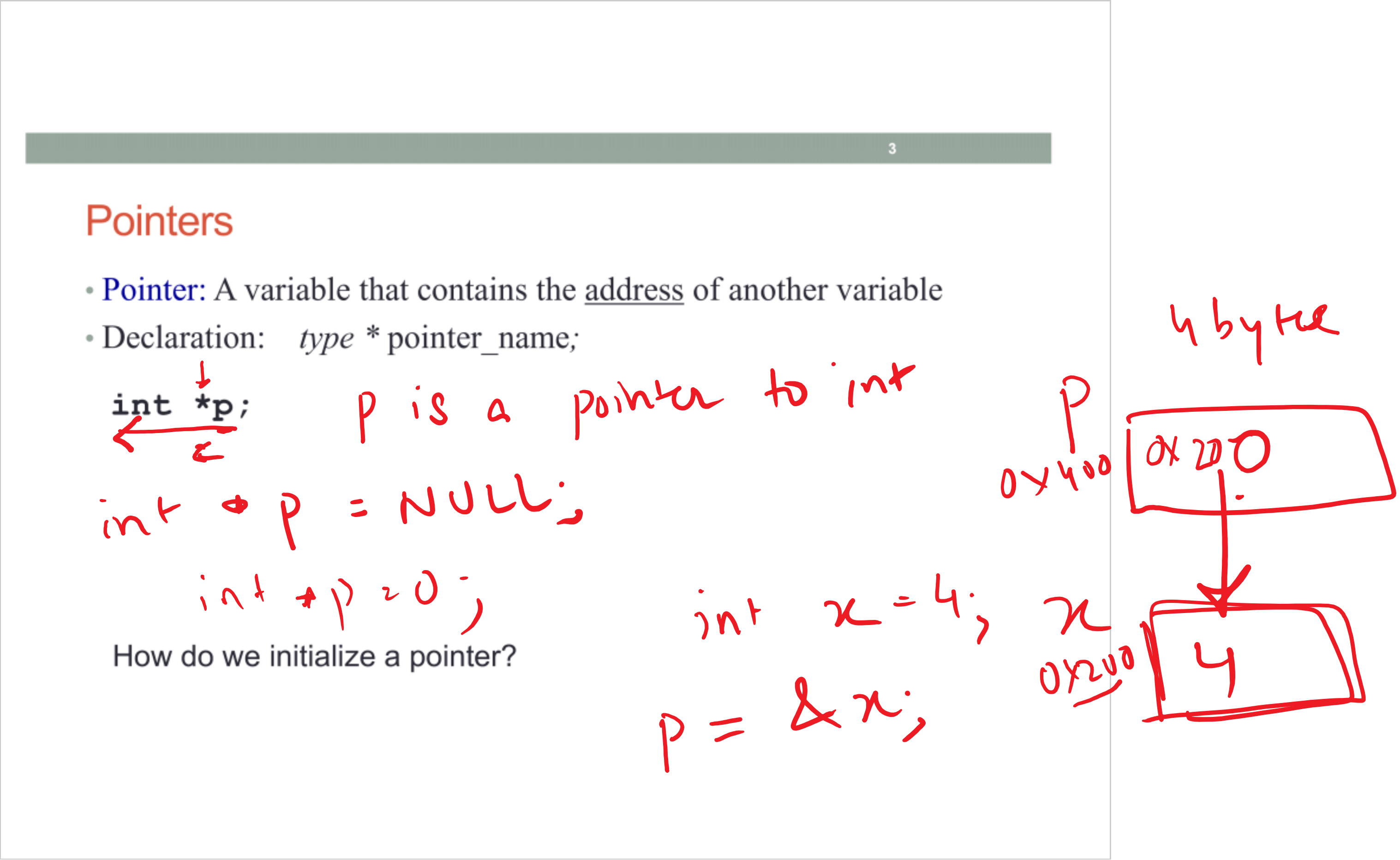


Announcements

- Midterm on Wed 04/26
- Study session today (04/23) from 7pm to 9pm in HFH 1132

Page 3

Monday, April 24, 2017 4:34 PM



Pointers

- Pointer: A variable that contains the address of another variable
- Declaration: *type* * pointer_name;

```
int *p;
```

p is a pointer to int

int *p = NULL;

int *p = 0;

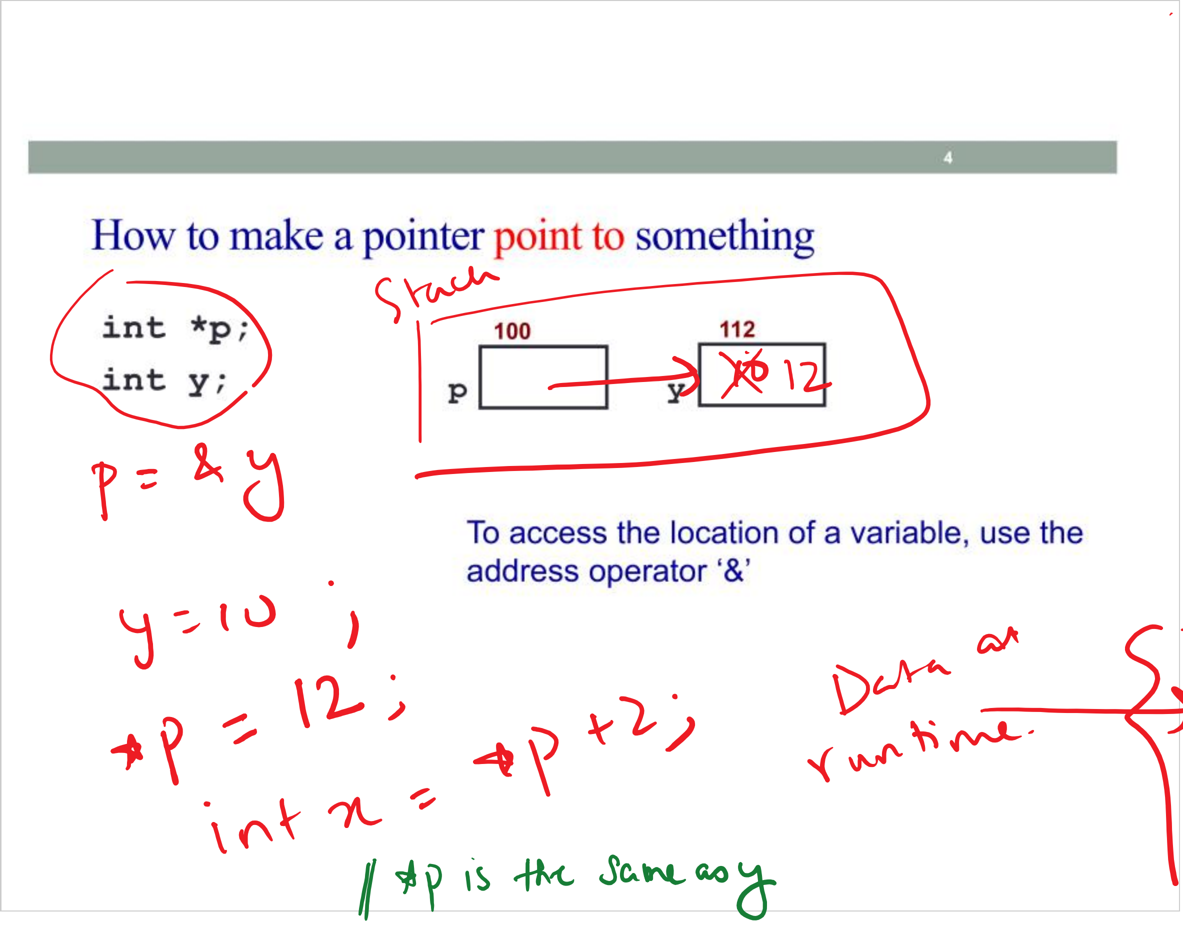
How do we initialize a pointer?

int x = 4;

p = &x;

Page 4

Monday, April 24, 2017 4:34 PM



How to make a pointer point to something

```
int *p;
int y;
```

To access the location of a variable, use the address operator '&'

p = &y

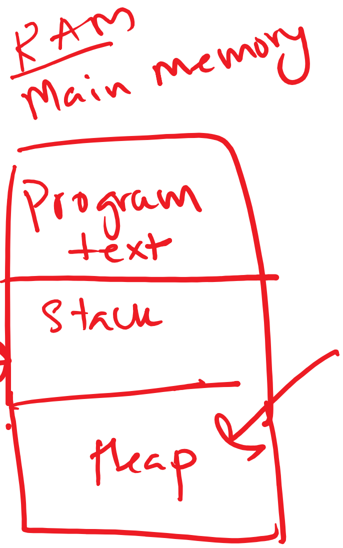
y = 10 ;

*p = 12;

int x = *p + 2;

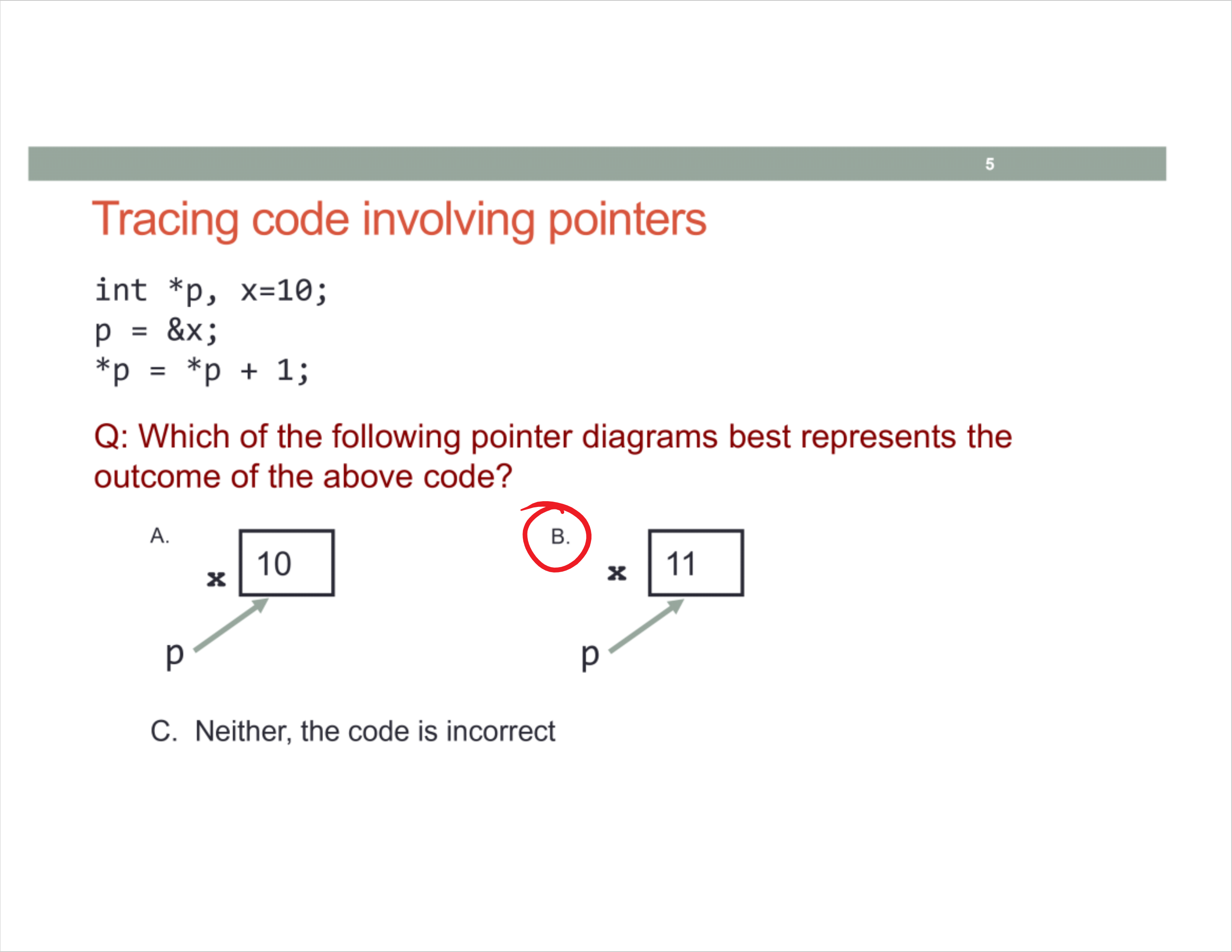
// *p is the same as y

Data at runtime.



Page 5

Monday, April 24, 2017 4:34 PM



5

Tracing code involving pointers

```
int *p, x=10;
p = &x;
*p = *p + 1;
```

Q: Which of the following pointer diagrams best represents the outcome of the above code?

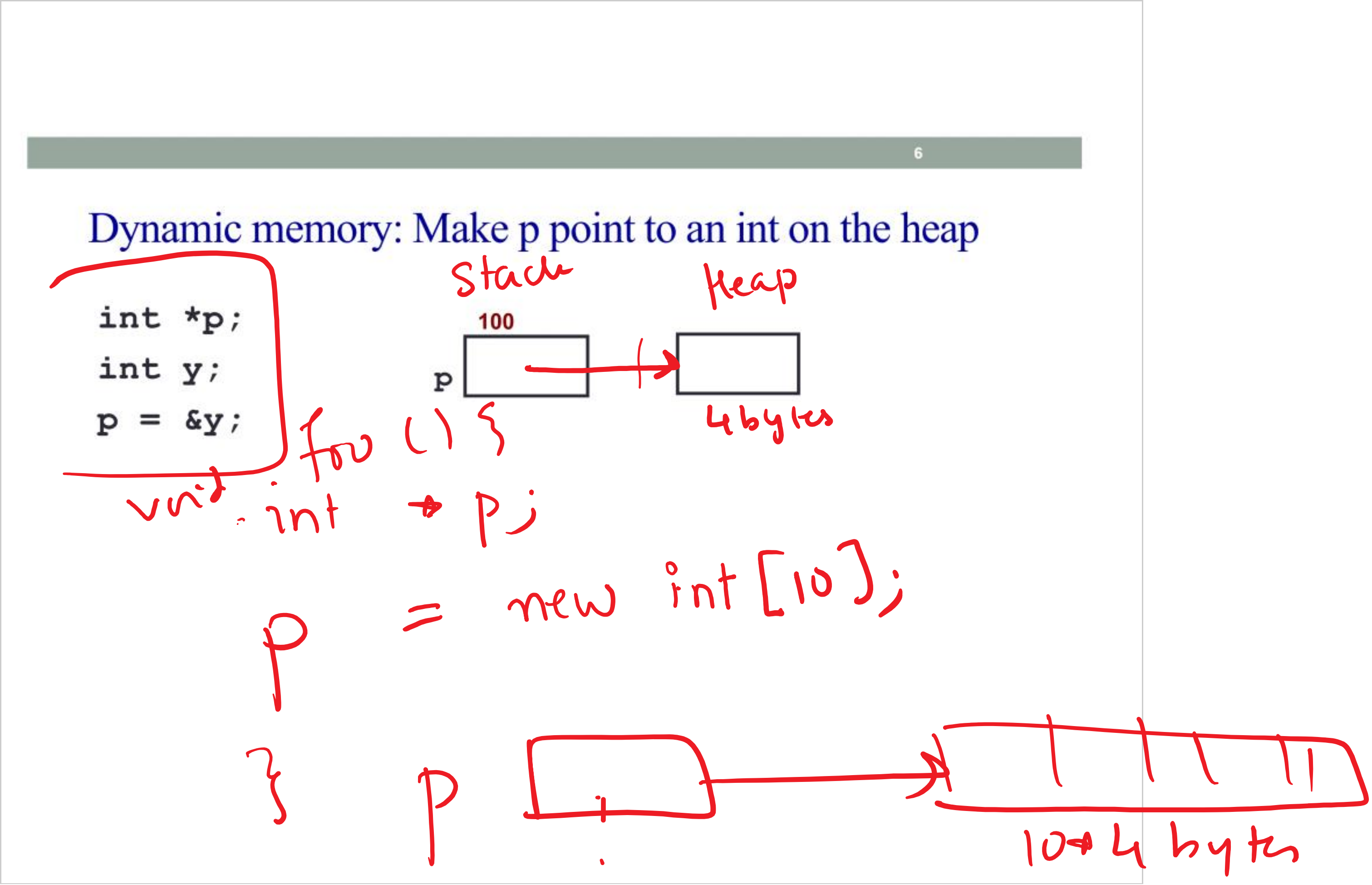
A.

[✓ B.]

C. Neither, the code is incorrect

Page 6

Monday, April 24, 2017 4:34 PM



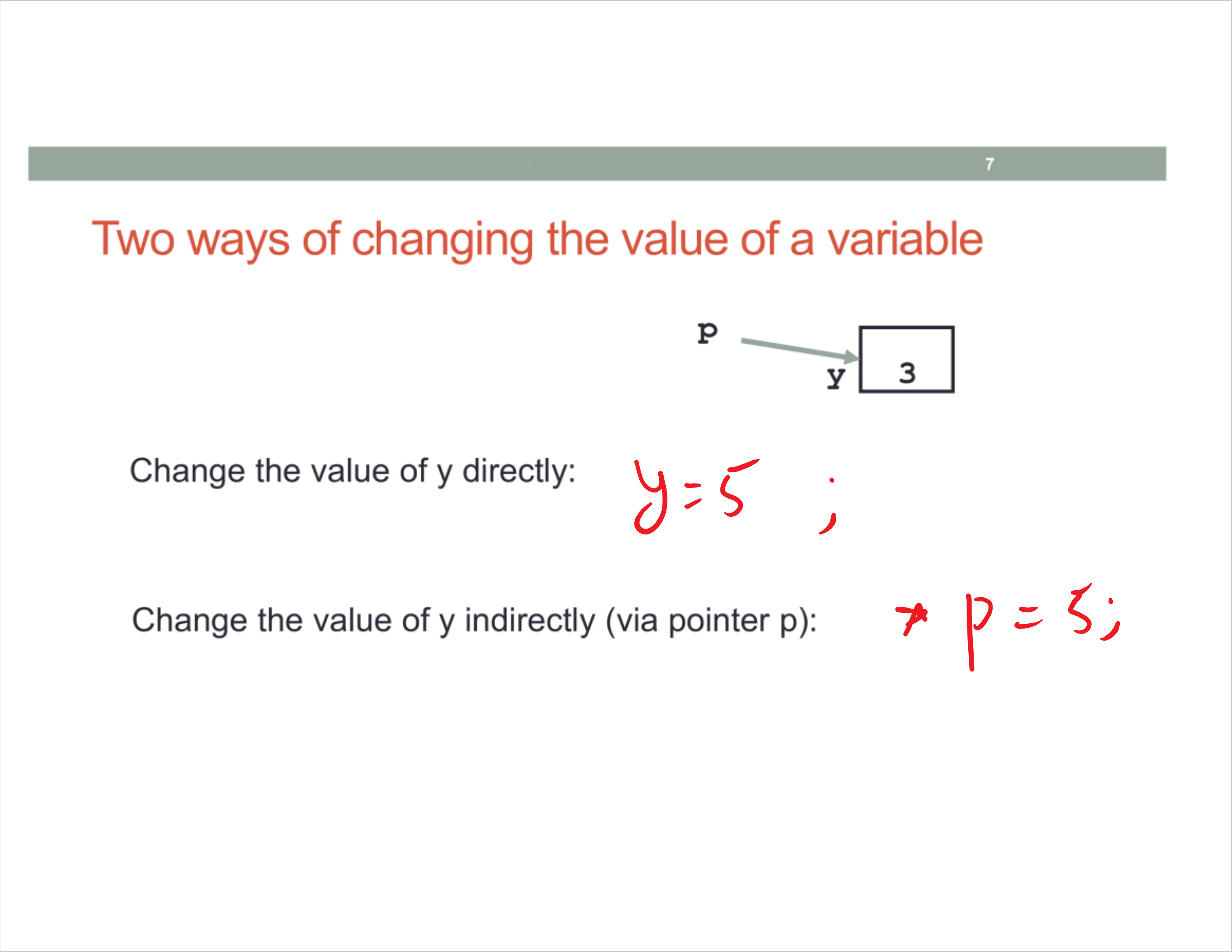
Dynamic memory: Make p point to an int on the heap

```
int *p;
int y;
p = &y;
```

```
void foo() {
    int * p;
    p = new int[10];
}
```

Page 7

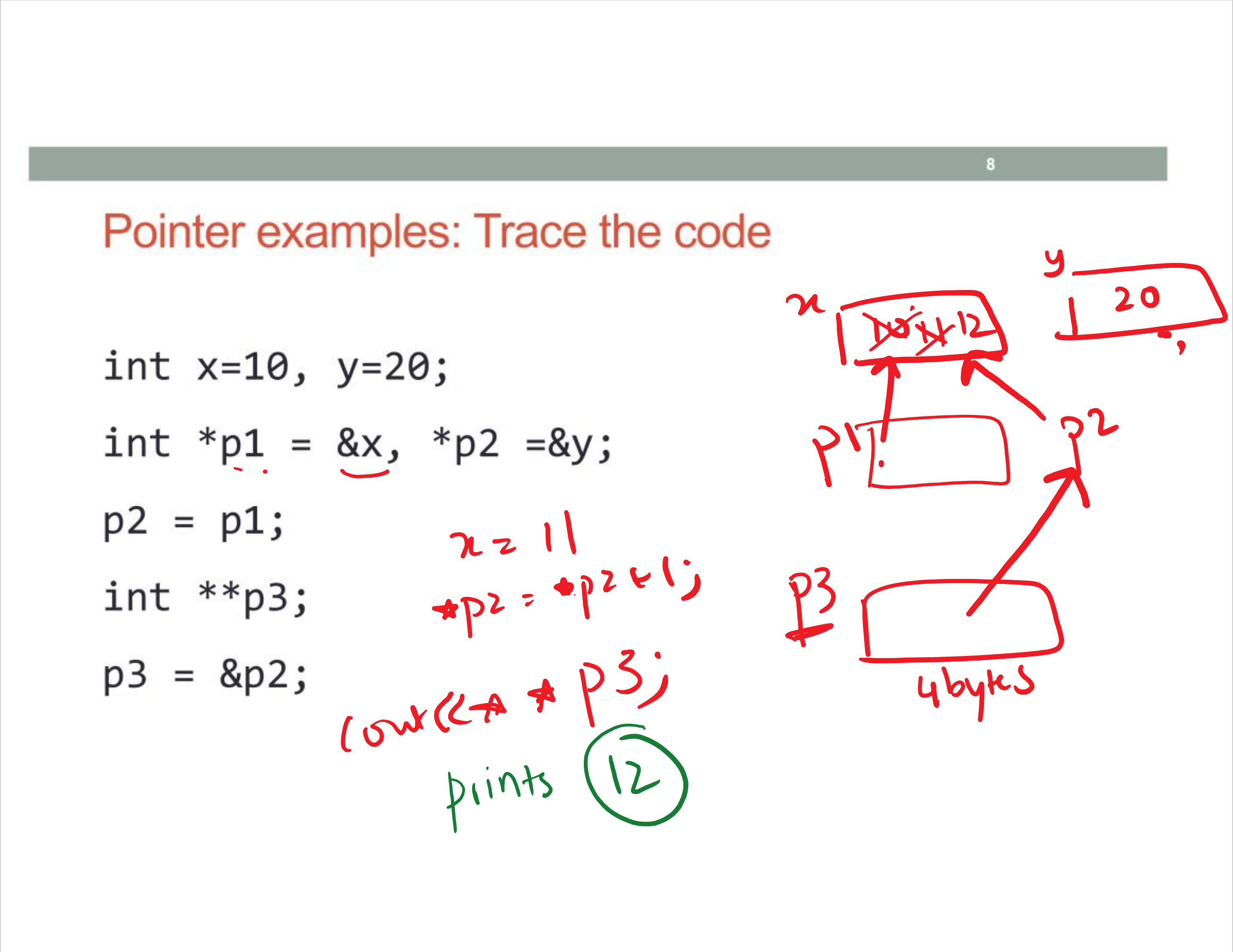
Monday, April 24, 2017 4:34 PM



Two ways of changing the value of a variable

Change the value of y directly: y=5 ;

Change the value of y indirectly (via pointer p): *p=5;



Pointer examples: Trace the code

```
int x=10, y=20;
int *p1 = &x, *p2 =&y;
p2 = p1;
int **p3;
p3 = &p2;
```

x = 11

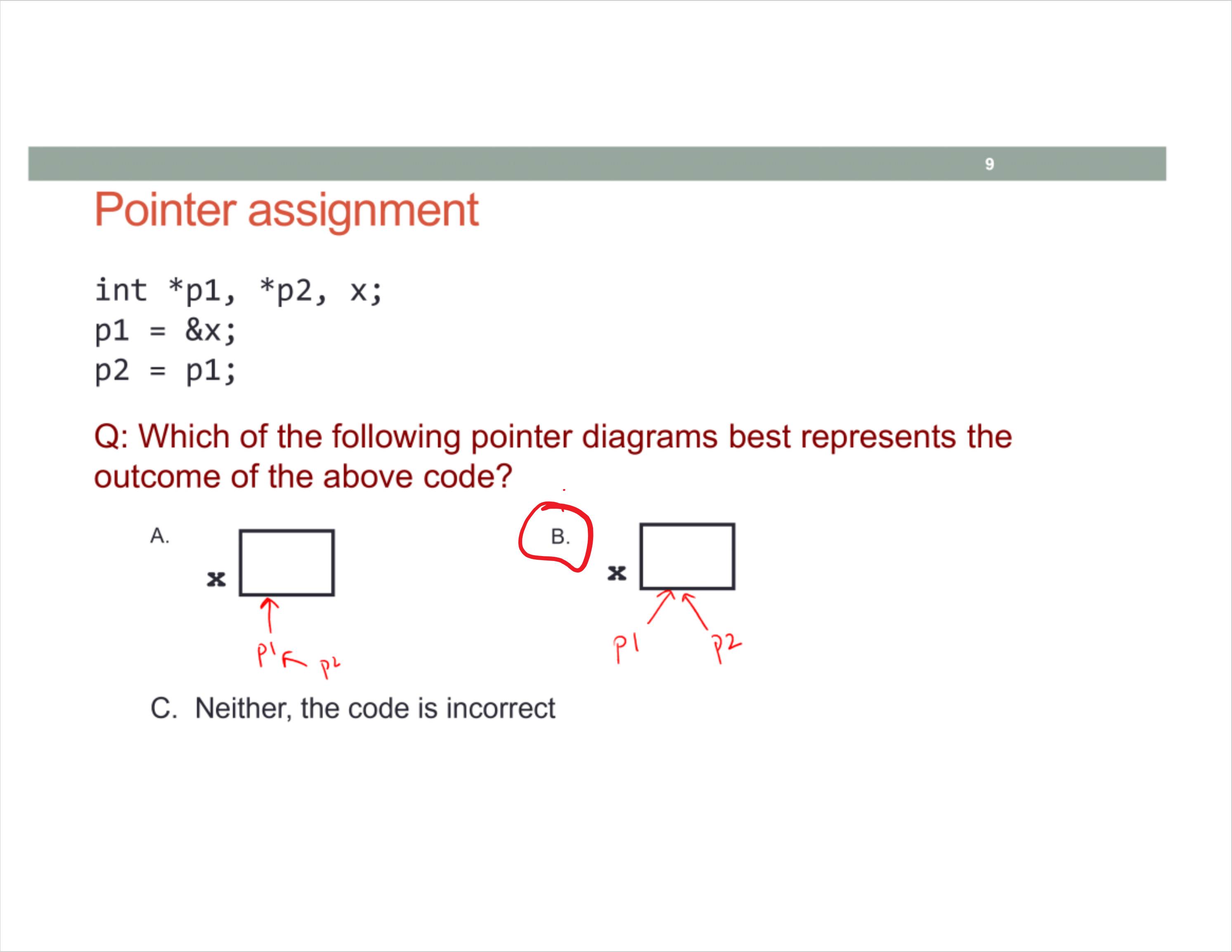
*p2 = *p2 + 1;

cout<< * p3;

prints 12

Page 9

Monday, April 24, 2017 4:34 PM



Pointer assignment

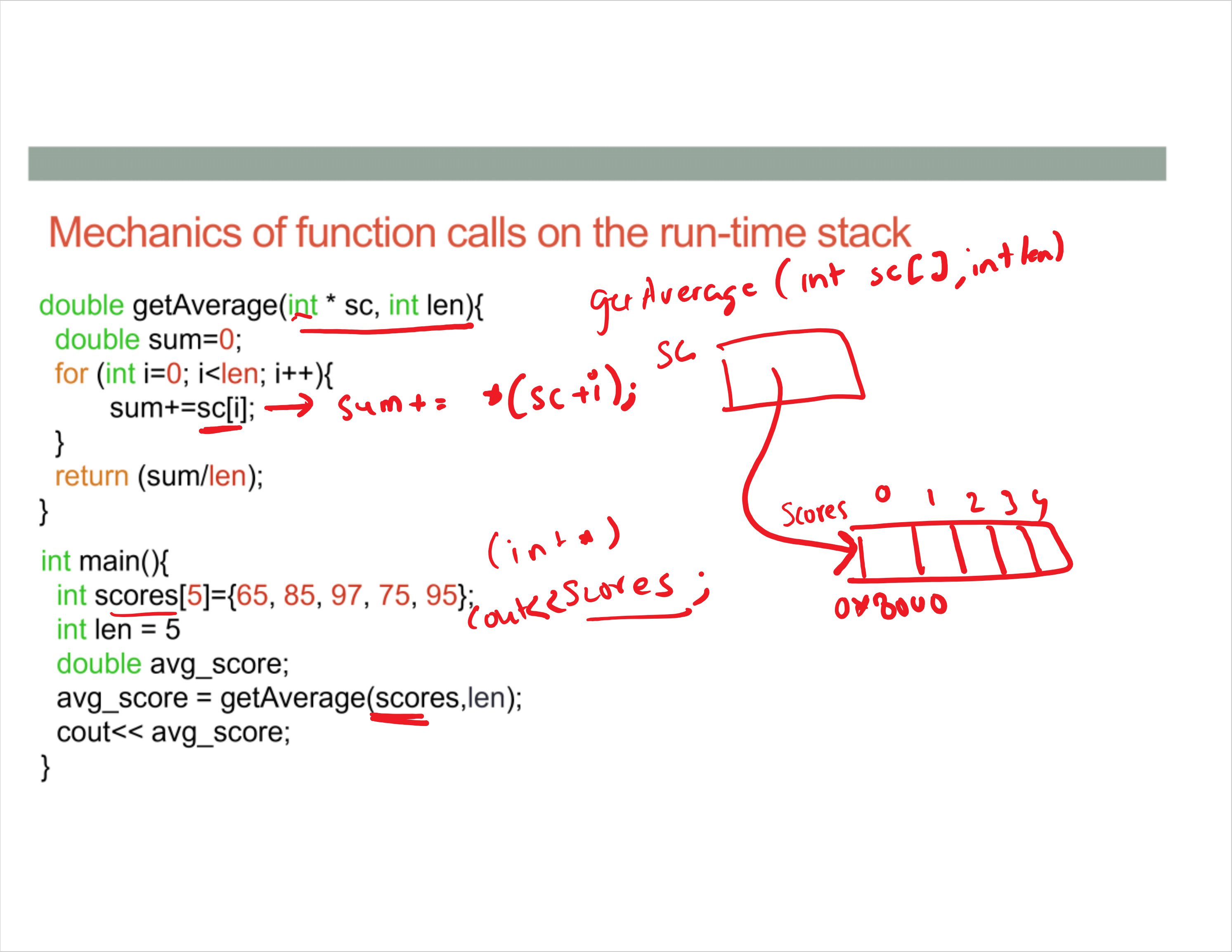
```
int *p1, *p2, x;
p1 = &x;
p2 = p1;
```

Q: Which of the following pointer diagrams best represents the outcome of the above code?

A.

[✓ B.]

C. Neither, the code is incorrect



Mechanics of function calls on the run-time stack

```
double getAverage(int * sc, int len){
  double sum=0;
  for (int i=0; i<len; i++){
        sum+=sc[i];
  }
  return (sum/len);
}
int main(){
  int scores[5]={65, 85, 97, 75, 95};
  int len = 5
  double avg_score;
  avg_score = getAverage(scores,len);
  cout<< avg_score;
}
```

getAverage(int sc[],int len)

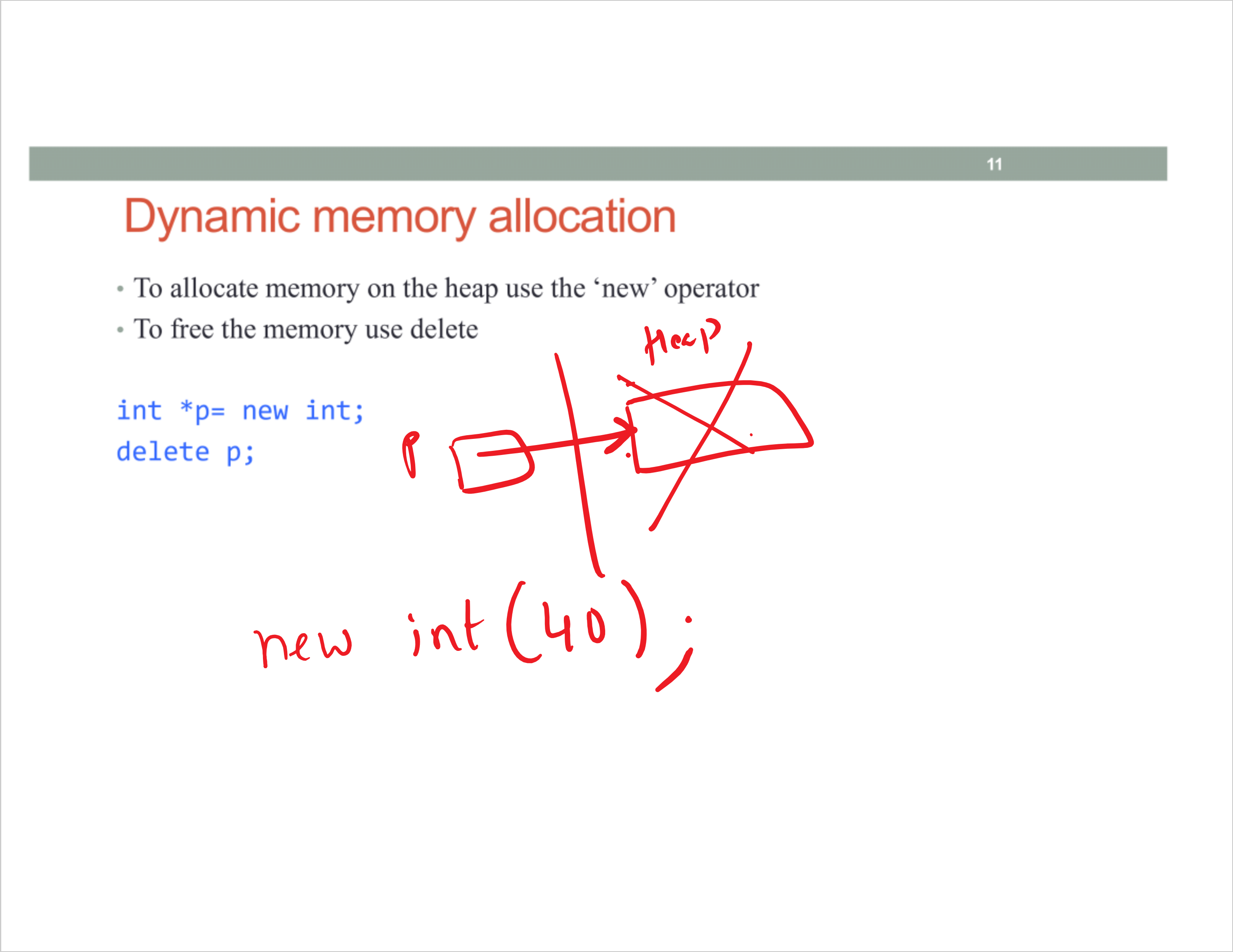
→ sum+= *(sc+i);

(int*)

cout<<Scores;

Page 11

Monday, April 24, 2017 4:35 PM



Dynamic memory allocation

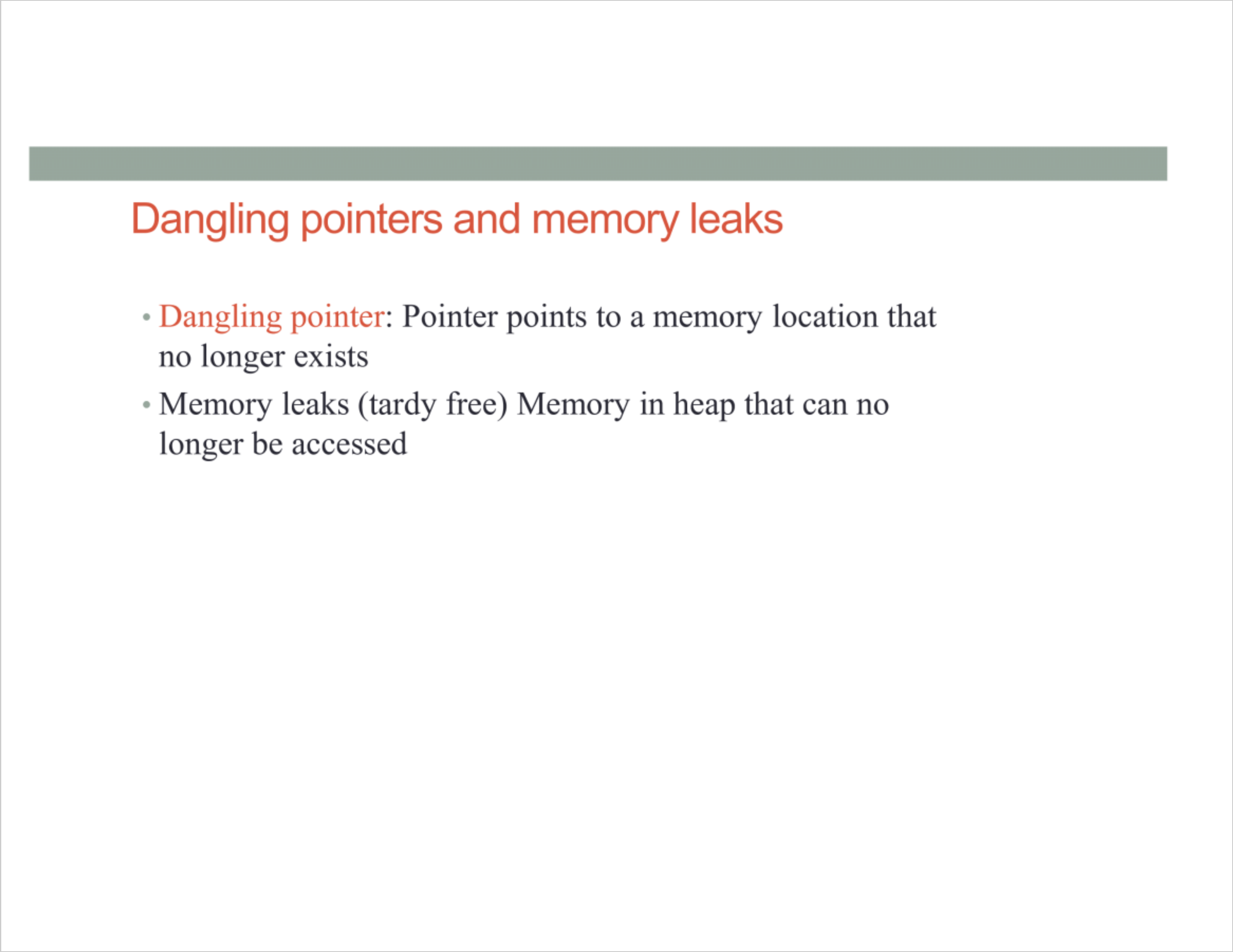
- To allocate memory on the heap use the ‘new’ operator
- To free the memory use delete

```
int *p= new int;
delete p;
```

new int(40);

Page 12

Monday, April 24, 2017 4:35 PM

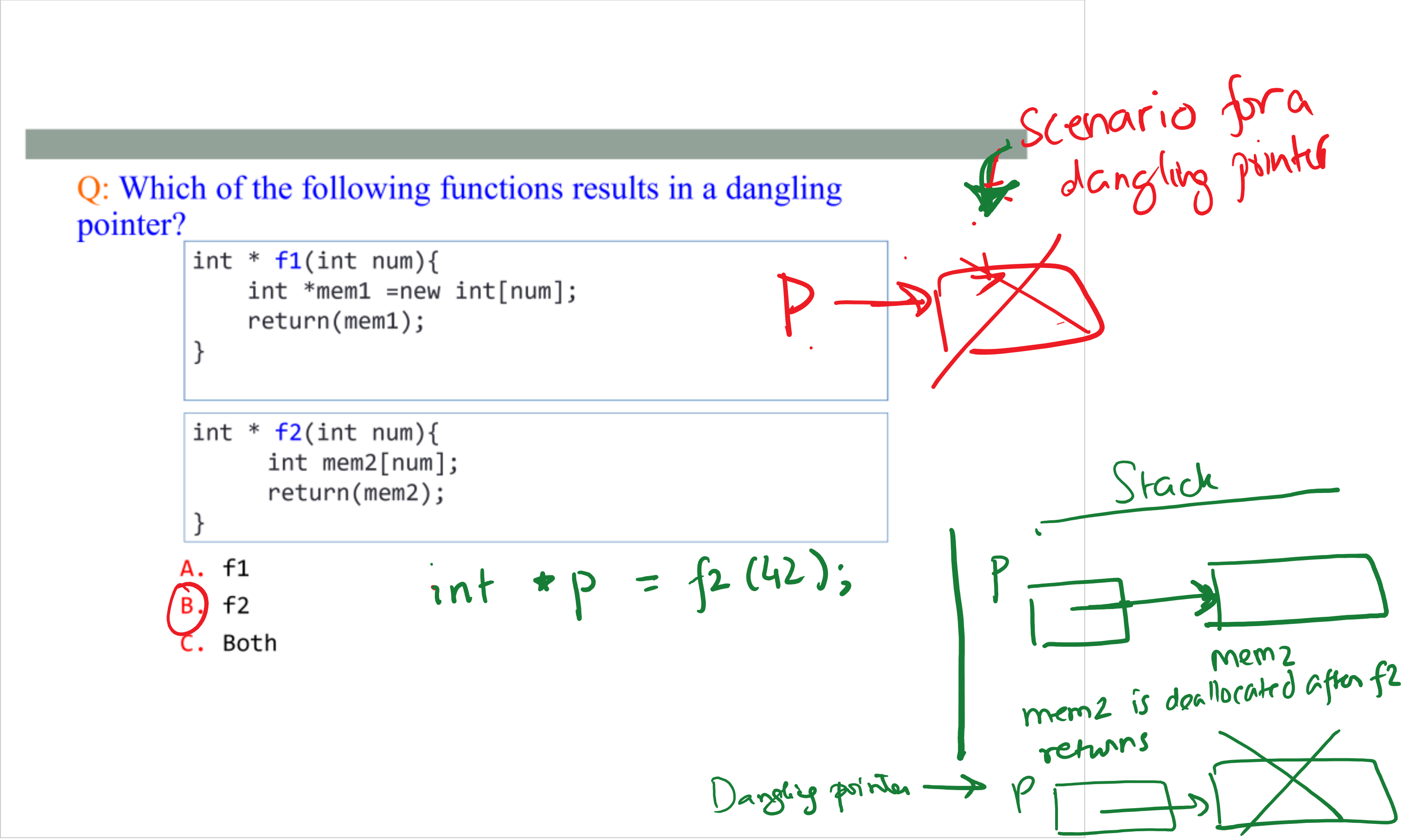


Dangling pointers and memory leaks

- Dangling pointer: Pointer points to a memory location that no longer exists
- Memory leaks (tardy free) Memory in heap that can no longer be accessed

Page 13

Monday, April 24, 2017 4:35 PM



Q: Which of the following functions results in a dangling pointer?

```
int * f1(int num){
    int *mem1 =new int[num];
    return(mem1);
}
```

```
int * f2(int num){
     int mem2[num];
     return(mem2);
}
```

A. f1

[✓ B. f2]

C. Both

Scenario for a dangling pointer

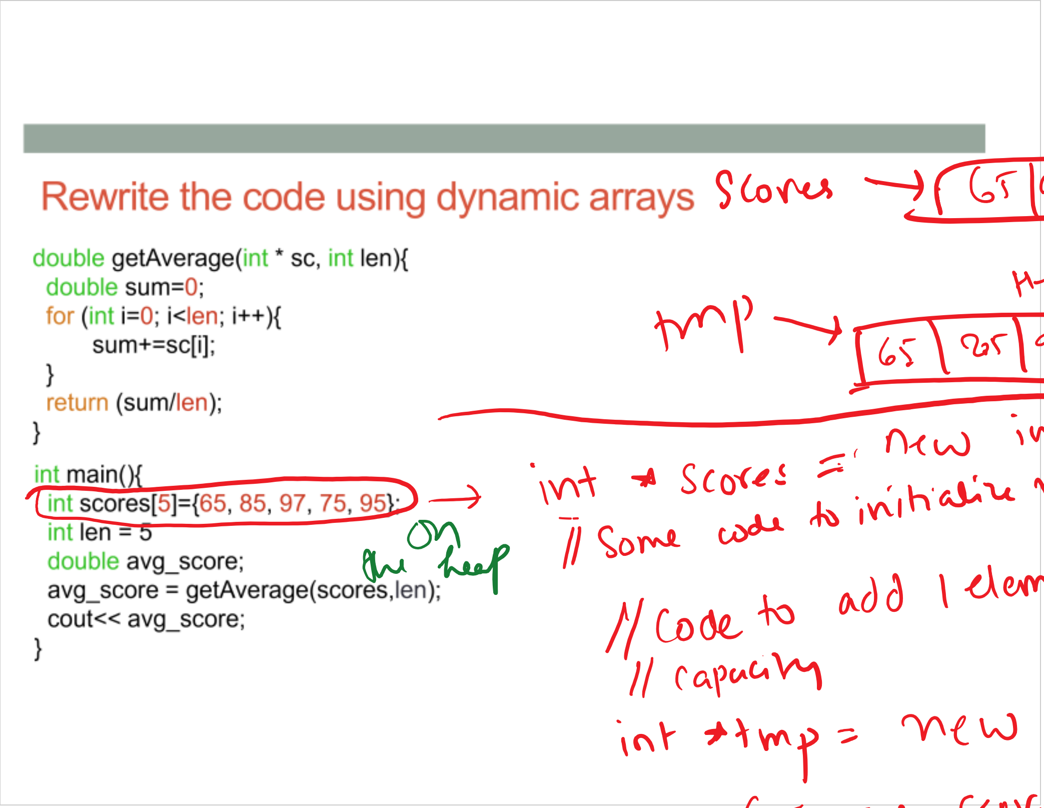
int *p = f2(42);

mem2 is deallocated after f2 returns

Dangling pointer → p

Page 14

Monday, April 24, 2017 4:35 PM



Rewrite the code using dynamic arrays

```
double getAverage(int * sc, int len){
  double sum=0;
  for (int i=0; i<len; i++){
       sum+=sc[i];
  }
  return (sum/len);
}

int main(){
  int scores[5]={65, 85, 97, 75, 95};
  int len = 5
  double avg_score;
  avg_score = getAverage(scores,len);
  cout<< avg_score;
}
```

On the heap

Scores → | 65 | 85 | 92 | 75 | 95 |

HEAD

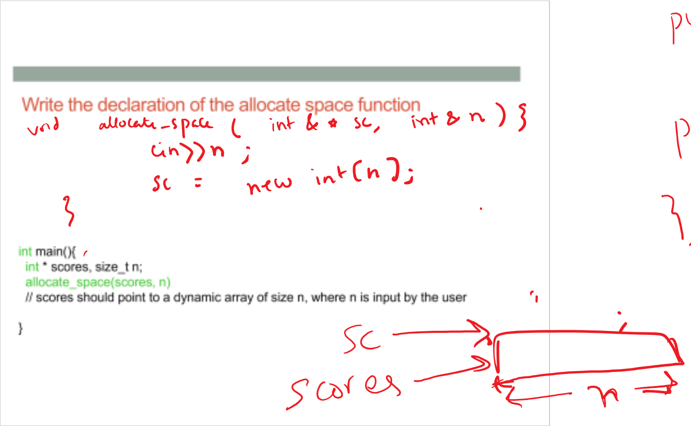
tmp → | 65 | 85 | 92 | 75 | 95. | |

```
int * scores = new int[5].
// Some code to initialize values

// Code to add 1 element more than the current
// capacity
int *tmp = new int[6];
copy(scores, scores+5, tmp);
delete [] scores;
scores = tmp
```

Page 15

Monday, April 24, 2017 4:35 PM



Write the declaration of the allocate space function

```
vnd   allocate_space (  int & * sc,   int & n ) {
            cin>>n ;
            sc  =   new  int[n];
   }
```

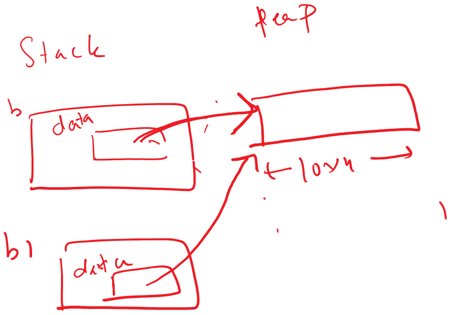
```
int main(){
  int * scores, size_t n;
  allocate_space(scores, n)
  // scores should point to a dynamic array of size n, where n is input by the user

}
```

```
class  bag {
  public:  // No copy constructor provided
           . . .
  private:
           int data[30];
};
```

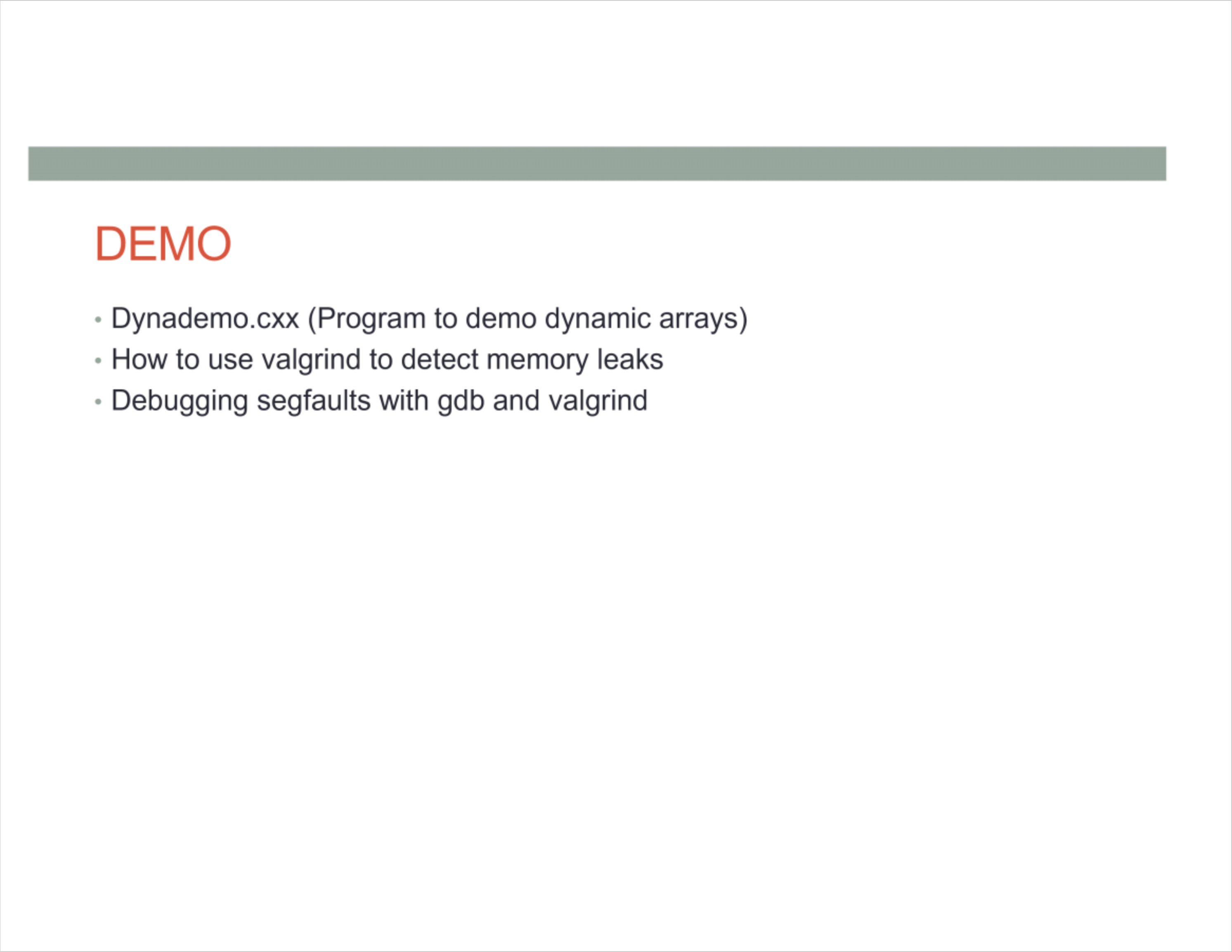
```
bag  b(10);
bag  b1(b);
```

if the default copy constructor was used → Shallow copy



Page 16

Monday, April 24, 2017 4:35 PM

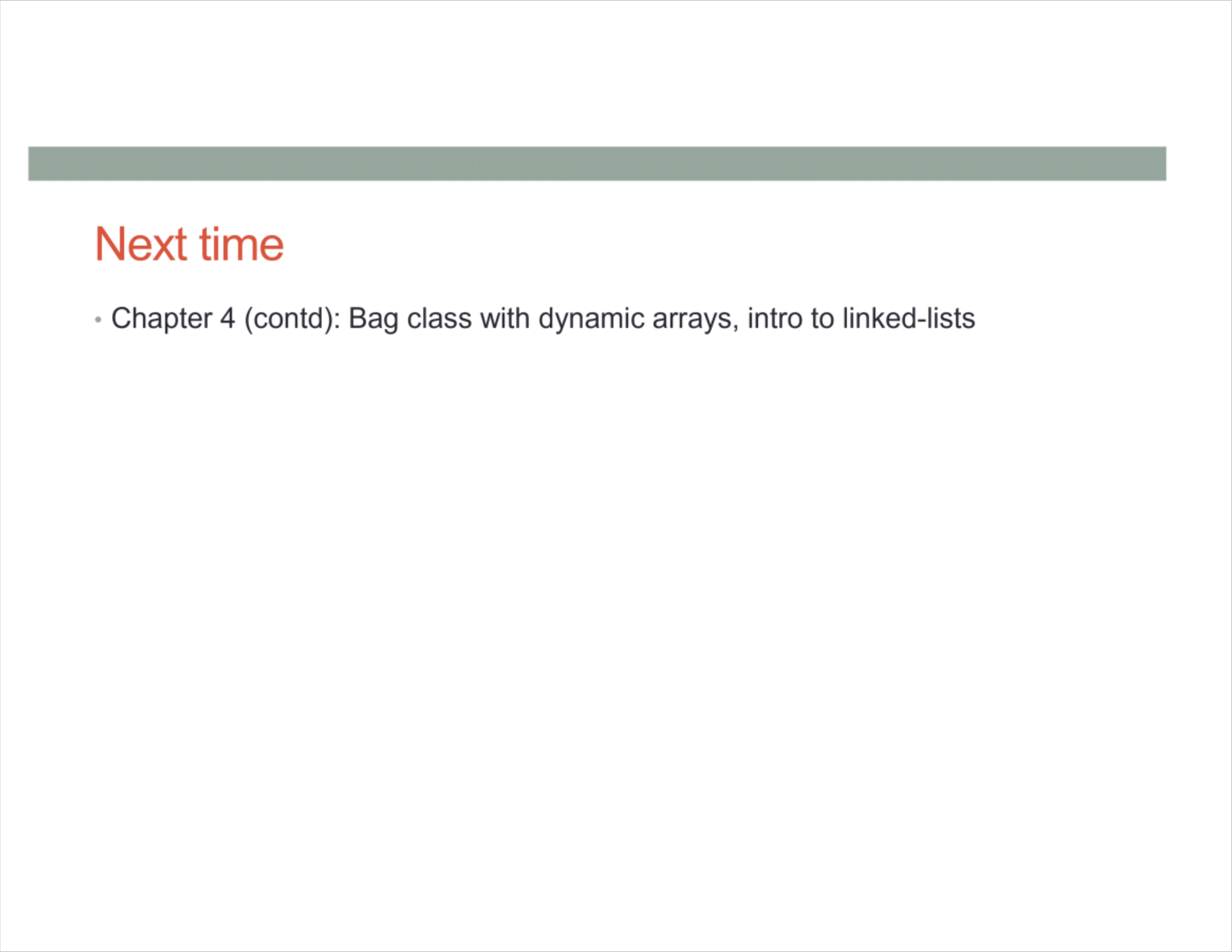


DEMO

- Dynademo.cxx (Program to demo dynamic arrays)
- How to use valgrind to detect memory leaks
- Debugging segfaults with gdb and valgrind

Page 17

Monday, April 24, 2017 4:35 PM



Next time

- Chapter 4 (contd): Bag class with dynamic arrays, intro to linked-lists