Pointers, Memory, and the Free Store

EECS 230

Spring 2016

00	10	20	30	40
01	11	21	31	41
02	12	22	32	42
03	13	23	33	43
04	14	24	34	44
05	15	25	35	45
06	16	26	36	46
07	17	27	37	47
08	18	28	38	48
09	19	29	39	49

	0_	1_	2_	3_	4_
0	00	10	20	30	40
1	01	11	21	31	41
2	02	12	22	32	42
3	03	13	23	33	43
4	04	14	24	34	44
5	05	15	25	35	45
6	06	16	26	36	46
7	07	17	27	37	47
8	08	18	28	38	48
9	09	19	29	39	49

	0_	1_	2_	3_	4_
0					
1					
2					
3					
4					
5					
6					
7					
8					
9					

Understanding memory

	0_	1_	2_
0	17	798	13
1	5	-4	0
2	0	50	-1
3	0	12	-1
4	65	2	-1
5	98	4	-1
6	99	6	87
7	20	8	4
8	66	10	16
9	0	-9	255

Understanding memory

	0_	1_	2_
0	17	798	13
1	5	-4	0
2	0	50	-1
3	0	12	-1
4	65	2	-1
5	98	4	-1
6	99	6	87
7	20	8	4
8	66	10	16
9	0	-9	255

Understanding memory

	0_	1_	2_
0	17	798	13
1	5	−4	0
2	0	50	−1
3	0	12	−1
4	65	2	−1
5	98	4	−1
6	99	6	87
7	20	8	4
8	66	10	16
9	0	−9	255

```
int x = 50;
```

Understanding memory

	0_	1_	2_
0	17	798	13
1	5	-4	0
2	0	50	-1
3	0	12	-1
4	65	2	-1
5	98	4	-1
6	99	6	87
7	20	8	4
8	66	10	16
9	0	-9	255

```
int x = 50;
// int x @ 12
```

Understanding memory

	0_	1_	2_
0	17	798	13
1	5	-4	0
2	0	50	-1
3	0	12	-1
4	65	2	-1
5	98	4	-1
6	99	6	87
7	20	8	4
8	66	10	16
9	0	-9	255

```
int x = 50;
// int x @ 12

int* px = &x;
```

Understanding memory

	0_	1_	2_
0	17	798	13
1	5	-4	0
2	0	50	-1
3	0	12	-1
4	65	2	-1
5	98	4	-1
6	99	6	87
7	20	8	4
8	66	10	16
9	0	-9	255

```
int x = 50;
// int x @ 12

int* px = &x;
// int* px @ 13
```

Understanding memory

	0_	1_	2_
0	17	798	13
1	5	-4	0
2	0	50	-1
3	0	12	-1
4	65	2	-1
5	98	4	-1
6	99	6	87
7	20	8	4
8	66	10	16
9	0	-9	255

```
int x = 50;
// int x @ 12

int* px = &x;
// int* px @ 13
```

Understanding memory

	0_	1_	2_
0	17	798	13
1	5	−4	0
2	0	50	−1
3	0	12	−1
4	65	2	−1
5	98	4	−1
6	99	6	87
7	20	8	4
8	66	10	16
9	0	−9	255

```
int x = 50;
// int x @ 12

int* px = &x;
// int* px @ 13

int a[] = { 2, 4, 6, 8, 10 };
```

Understanding memory

	0_	1_	2_
0	17	798	13
1	5	-4	0
2	0	50	-1
3	0	12	-1
4	65	2	-1
5	98	4	-1
6	99	6	87
7	20	8	4
8	66	10	16
9	0	-9	255

```
int x = 50;
// int x @ 12

int* px = &x;
// int* px @ 13

int a[] = { 2, 4, 6, 8, 10 };
// int a[5] @ 14
```

Understanding memory

	0_	1_	2_
0	17	798	13
1	5	-4	0
2	0	50	-1
3	0	12	-1
4	65	2	-1
5	98	4	-1
6	99	6	87
7	20	8	4
8	66	10	16
9	0	-9	255

```
int x = 50;
// int x @ 12

int* px = &x;
// int* px @ 13

int a[] = { 2, 4, 6, 8, 10 };
// int a[5] @ 14

int** ppx = &px;
```

Understanding memory

	0_	1_	2_
0	17	798	13
1	5	−4	0
2	0	50	−1
3	0	12	−1
4	65	2	−1
5	98	4	−1
6	99	6	87
7	20	8	4
8	66	10	16
9	0	−9	255

```
int x = 50;
// int x @ 12

int* px = &x;
// int* px @ 13

int a[] = { 2, 4, 6, 8, 10 };
// int a[5] @ 14

int** ppx = &px;
// int** ppx @ 20
```

Understanding memory

	0_	1_	2_
0	17	798	13
1	5	−4	0
2	0	50	−1
3	0	12	−1
4	65	2	−1
5	98	4	−1
6	99	6	87
7	20	8	4
8	66	10	16
9	0	−9	255

```
int x = 50;
// int x @ 12

int* px = &x;
// int* px @ 13

int a[] = { 2, 4, 6, 8, 10 };
// int a[5] @ 14

int** ppx = &px;
// int** ppx @ 20
```

Understanding memory

	0_	1_	2_
0	17	798	13
1	5	−4	0
2	0	50	−1
3	0	12	−1
4	65	2	−1
5	98	4	−1
6	99	6	87
7	20	8	4
8	66	10	16
9	0	−9	255

```
int x = 50;
// int x @ 12

int* px = &x;
// int* px @ 13

int a[] = { 2, 4, 6, 8, 10 };
// int a[5] @ 14

int** ppx = &px;
// int** ppx @ 20
```

Execution on the stack

```
int g(int x)
{
    return x + 2;
}

int f(int a, int b)
{
    return a * b;
}

int main()
{
    cout << f(g(3), g(8));
}
```

	0_	1_	2_
0			
1			
2			
3			
4			
5			
6			
7			
8			
9			

Execution on the stack

```
int g(int x)
{          @ 4
    return x + 2;
}    @ 3

int f(int a, int b)
{         @ 4   @ 5
    return a * b;
}    @ 3

int main()
{                    @ 1   @ 2
    cout << f(g(3), g(8));
}                  @ 0
```

	0_	1_	2_
0			
1			
2			
3			
4	8		
5			
6			
7			
8			
9			

Execution on the stack

```
int g(int x)
{          @ 4
    return x + 2;
}     @ 3

int f(int a, int b)
{          @ 4   @ 5
    return a * b;
}     @ 3

int main()
{                  @ 1   @ 2
    cout << f(g(3), g(8));
}                @ 0
```

	0_	1_	2_
0			
1			
2			
3	10		
4	8		
5			
6			
7			
8			
9			

Execution on the stack

```
int g(int x)
{        @ 4
    return x + 2;
}    @ 3

int f(int a, int b)
{        @ 4   @ 5
    return a * b;
}    @ 3

int main()
{                  @ 1   @ 2
    cout << f(g(3), g(8));
}                @ 0
```

	0_	1_	2_
0			
1			
2	10		
3	10		
4	8		
5			
6			
7			
8			
9			

Execution on the stack

```
int g(int x)
{         @ 4
    return x + 2;
}    @ 3

int f(int a, int b)
{        @ 4   @ 5
    return a ∗ b;
}    @ 3

int main()
{                   @ 1   @ 2
    cout << f(g(3), g(8));
}                 @ 0
```

	0_	1_	2_
0			
1			
2	10		
3	10		
4	3		
5			
6			
7			
8			
9			

Execution on the stack

```
int g(int x)
{           @ 4
     return x + 2;
}      @ 3

int f(int a, int b)
{           @ 4   @ 5
     return a ∗ b;
}      @ 3

int main()
{                       @ 1   @ 2
     cout << f(g(3), g(8));
}                    @ 0
```

	0_	1_	2_
0			
1			
2	10		
3	5		
4	3		
5			
6			
7			
8			
9			

Execution on the stack

```
int g(int x)
{         @ 4
    return x + 2;
}    @ 3

int f(int a, int b)
{        @ 4   @ 5
    return a * b;
}    @ 3

int main()
{                   @ 1   @ 2
    cout << f(g(3), g(8));
}                @ 0
```

	0_	1_	2_
0			
1	5		
2	10		
3	5		
4	3		
5			
6			
7			
8			
9			

Execution on the stack

```
int g(int x)
{          @ 4
     return x + 2;
}     @ 3

int f(int a, int b)
{          @ 4   @ 5
     return a * b;
}     @ 3

int main()
{                    @ 1   @ 2
     cout << f(g(3), g(8));
}                   @ 0
```

	0_	1_	2_
0			
1	5		
2	10		
3	5		
4	5		
5	10		
6			
7			
8			
9			

Execution on the stack

```
int g(int x)
{         @ 4
    return x + 2;
}    @ 3

int f(int a, int b)
{         @ 4   @ 5
    return a * b;
}    @ 3

int main()
{                  @ 1   @ 2
    cout << f(g(3), g(8));
}                  @ 0
```

	0_	1_	2_
0			
1	5		
2	10		
3	50		
4	5		
5	10		
6			
7			
8			
9			

Execution on the stack

```
int g(int x)
{        @ 4
    return x + 2;
}    @ 3

int f(int a, int b)
{        @ 4   @ 5
    return a * b;
}    @ 3

int main()
{                   @ 1   @ 2
    cout << f(g(3), g(8));
}                 @ 0
```

	0_	1_	2_
0	50		
1	5		
2	10		
3	50		
4	5		
5	10		
6			
7			
8			
9			

Execution on the stack (more realistic)

```
int g(int x)
{           @ 4
    return x + 2;
}     @ 3

int f(int a, int b)
{          @ 4   @ 5
    return a * b;
}     @ 3

int main()
{                      @ 1   @ 2
    cout << f(g(3), g(8));
}                    @ 0
```

	0_	1_	2_
0	17	798	13
1	5	−4	0
2	0	50	−1
3	0	12	−1
4	65	2	−1
5	98	4	−1
6	99	6	87
7	20	8	4
8	66	10	16
9	0	−9	255

Execution on the stack (more realistic)

```
int g(int x)
{         @ 4
    return x + 2;
}    @ 3

int f(int a, int b)
{        @ 4   @ 5
    return a * b;
}    @ 3

int main()
{                     @ 1    @ 2
    cout << f(g(3), g(8));
}                  @ 0
```

	0_	1_	2_
0	17	798	13
1	5	−4	0
2	0	50	−1
3	0	12	−1
4	8	2	−1
5	98	4	−1
6	99	6	87
7	20	8	4
8	66	10	16
9	0	−9	255

Execution on the stack (more realistic)

```
int g(int x)
{         @ 4
    return x + 2;
}     @ 3

int f(int a, int b)
{        @ 4   @ 5
    return a * b;
}     @ 3

int main()
{                    @ 1   @ 2
    cout << f(g(3), g(8));
}                   @ 0
```

	0_	1_	2_
0	17	798	13
1	5	−4	0
2	0	50	−1
3	10	12	−1
4	8	2	−1
5	98	4	−1
6	99	6	87
7	20	8	4
8	66	10	16
9	0	−9	255

Execution on the stack (more realistic)

```
int g(int x)
{        @ 4
    return x + 2;
}    @ 3

int f(int a, int b)
{        @ 4   @ 5
    return a * b;
}    @ 3

int main()
{                   @ 1   @ 2
    cout << f(g(3), g(8));
}                  @ 0
```

	0_	1_	2_
0	17	798	13
1	5	−4	0
2	10	50	−1
3	10	12	−1
4	8	2	−1
5	98	4	−1
6	99	6	87
7	20	8	4
8	66	10	16
9	0	−9	255

Execution on the stack (more realistic)

```
int g(int x)
{         @ 4
    return x + 2;
}    @ 3

int f(int a, int b)
{         @ 4  @ 5
    return a * b;
}    @ 3

int main()
{                   @ 1  @ 2
    cout << f(g(3), g(8));
}                  @ 0
```

	0_	1_	2_
0	17	798	13
1	5	−4	0
2	10	50	−1
3	10	12	−1
4	3	2	−1
5	98	4	−1
6	99	6	87
7	20	8	4
8	66	10	16
9	0	−9	255

Execution on the stack (more realistic)

```
int g(int x)
{         @ 4
     return x + 2;
}    @ 3

int f(int a, int b)
{        @ 4   @ 5
     return a * b;
}    @ 3

int main()
{                   @ 1   @ 2
     cout << f(g(3), g(8));
}                 @ 0
```

	0_	1_	2_
0	17	798	13
1	5	−4	0
2	10	50	−1
3	5	12	−1
4	3	2	−1
5	98	4	−1
6	99	6	87
7	20	8	4
8	66	10	16
9	0	−9	255

Execution on the stack (more realistic)

```
int g(int x)
{          @ 4
    return x + 2;
}     @ 3

int f(int a, int b)
{         @ 4   @ 5
    return a * b;
}     @ 3

int main()
{                   @ 1   @ 2
    cout << f(g(3), g(8));
}                 @ 0
```

	0_	1_	2_
0	17	798	13
1	5	−4	0
2	10	50	−1
3	5	12	−1
4	3	2	−1
5	98	4	−1
6	99	6	87
7	20	8	4
8	66	10	16
9	0	−9	255

Execution on the stack (more realistic)

```
int g(int x)
{          @ 4
    return x + 2;
}    @ 3

int f(int a, int b)
{          @ 4   @ 5
    return a * b;
}    @ 3

int main()
{                   @ 1   @ 2
    cout << f(g(3), g(8));
}                  @ 0
```

	0_	1_	2_
0	17	798	13
1	5	−4	0
2	10	50	−1
3	5	12	−1
4	5	2	−1
5	10	4	−1
6	99	6	87
7	20	8	4
8	66	10	16
9	0	−9	255

Execution on the stack (more realistic)

```
int g(int x)
{         @ 4
    return x + 2;
}     @ 3

int f(int a, int b)
{        @ 4   @ 5
    return a * b;
}     @ 3

int main()
{                    @ 1   @ 2
    cout << f(g(3), g(8));
}                  @ 0
```

	0_	1_	2_
0	17	798	13
1	5	−4	0
2	10	50	−1
3	50	12	−1
4	5	2	−1
5	10	4	−1
6	99	6	87
7	20	8	4
8	66	10	16
9	0	−9	255

Execution on the stack (more realistic)

```
int g(int x)
{        @ 4
    return x + 2;
}    @ 3

int f(int a, int b)
{        @ 4   @ 5
    return a * b;
}    @ 3

int main()
{                    @ 1   @ 2
    cout << f(g(3), g(8));
}                  @ 0
```

	0_	1_	2_
0	50	798	13
1	5	−4	0
2	10	50	−1
3	50	12	−1
4	5	2	−1
5	10	4	−1
6	99	6	87
7	20	8	4
8	66	10	16
9	0	−9	255

Recursion on the stack

fact(n)	n @	result @
fact(5)	1	0

```
int fact(int n)
{
   if (n == 0)
      return 1;
   else
      return n ∗ fact(n − 1)
}

      fact(5);
```

	0_	1_	2_
0			
1	5		
2			
3			
4			
5			
6			
7			
8			
9			

Recursion on the stack

fact(n)	n @	result @
fact(5)	1	0
fact(4)	3	2

```
int fact(int n)
{
   if (n == 0)
      return 1;
   else
      return n * fact(n - 1)
}

      fact(5);
```

	0_	1_	2_
0			
1	5		
2			
3	4		
4			
5			
6			
7			
8			
9			

Recursion on the stack

fact(n)	n @	result @
fact(5)	1	0
fact(4)	3	2
fact(3)	5	4

```
int fact(int n)
{
   if (n == 0)
      return 1;
   else
      return n * fact(n − 1)
}

      fact(5);
```

	0_	1_	2_
0			
1	5		
2			
3	4		
4			
5	3		
6			
7			
8			
9			

Recursion on the stack

fact(n)	n @	result @
fact(5)	1	0
fact(4)	3	2
fact(3)	5	4
fact(2)	7	6

```
int fact(int n)
{
  if (n == 0)
    return 1;
  else
    return n * fact(n - 1)
}

    fact(5);
```

	0_	1_	2_
0			
1	5		
2			
3	4		
4			
5	3		
6			
7	2		
8			
9			

Recursion on the stack

fact(n)	n @	result @
fact(5)	1	0
fact(4)	3	2
fact(3)	5	4
fact(2)	7	6
fact(1)	9	8

```
int fact(int n)
{
   if (n == 0)
      return 1;
   else
      return n * fact(n − 1)
}

      fact(5);
```

	0_	1_	2_
0			
1	5		
2			
3	4		
4			
5	3		
6			
7	2		
8			
9	1		

Recursion on the stack

fact(n)	n @	result @
fact(5)	1	0
fact(4)	3	2
fact(3)	5	4
fact(2)	7	6
fact(1)	9	8
fact(0)	11	10

```
int fact(int n)
{
   if (n == 0)
      return 1;
   else
      return n * fact(n - 1)
}

      fact(5);
```

	0_	1_	2_
0			
1	5	0	
2			
3	4		
4			
5	3		
6			
7	2		
8			
9	1		

Recursion on the stack

fact(n)	n @	result @
fact(5)	1	0
fact(4)	3	2
fact(3)	5	4
fact(2)	7	6
fact(1)	9	8
fact(0)	11	10

```
int fact(int n)
{
   if (n == 0)
      return 1;
   else
      return n * fact(n − 1)
}

      fact(5);
```

	0_	1_	2_
0		1	
1	5	0	
2			
3	4		
4			
5	3		
6			
7	2		
8			
9	1		

Recursion on the stack

fact(n)	n @	result @
fact(5)	1	0
fact(4)	3	2
fact(3)	5	4
fact(2)	7	6
fact(1)	9	8
fact(0)	11	10

```
int fact(int n)
{
    if (n == 0)
        return 1;
    else
        return n * fact(n − 1)
}

        fact(5);
```

	0_	1_	2_
0		1	
1	5	0	
2			
3	4		
4			
5	3		
6			
7	2		
8	1		
9	1		

Recursion on the stack

fact(n)	n @	result @
fact(5)	1	0
fact(4)	3	2
fact(3)	5	4
fact(2)	7	6
fact(1)	9	8
fact(0)	11	10

```
int fact(int n)
{
   if (n == 0)
      return 1;
   else
      return n * fact(n - 1)
}

      fact(5);
```

	0_	1_	2_
0		1	
1	5	0	
2			
3	4		
4			
5	3		
6	2		
7	2		
8	1		
9	1		

Recursion on the stack

fact(n)	n @	result @
fact(5)	1	0
fact(4)	3	2
fact(3)	5	4
fact(2)	7	6
fact(1)	9	8
fact(0)	11	10

```
int fact(int n)
{
   if (n == 0)
      return 1;
   else
      return n * fact(n - 1)
}

      fact(5);
```

	0_	1_	2_
0		1	
1	5	0	
2			
3	4		
4	6		
5	3		
6	2		
7	2		
8	1		
9	1		

Recursion on the stack

fact(n)	n @	result @
fact(5)	1	0
fact(4)	3	2
fact(3)	5	4
fact(2)	7	6
fact(1)	9	8
fact(0)	11	10

```
int fact(int n)
{
   if (n == 0)
      return 1;
   else
      return n * fact(n − 1)
}

      fact(5);
```

	0_	1_	2_
0		1	
1	5	0	
2	24		
3	4		
4	6		
5	3		
6	2		
7	2		
8	1		
9	1		

Recursion on the stack

fact(n)	n @	result @
fact(5)	1	0
fact(4)	3	2
fact(3)	5	4
fact(2)	7	6
fact(1)	9	8
fact(0)	11	10

```
int fact(int n)
{
   if (n == 0)
      return 1;
   else
      return n * fact(n - 1)
}

      fact(5);
```

	0_	1_	2_
0	120	1	
1	5	0	
2	24		
3	4		
4	6		
5	3		
6	2		
7	2		
8	1		
9	1		

Can’t return pointers to stack variables

This is fundamentally broken:

```
int* ptr_to_3()
{
	int x = 3;
	return &x;
}
```

Can’t return pointers to stack variables

This is fundamentally broken:

```
int* ptr_to_3()
{
    int x = 3;
    return &x;
}
```

So is this:

```
int* ptr_to_array()
{
    int x[] = { 3, 4, 5 };
    return &x;
}
```

The heap

```
int* p = new int{3};
```

The heap

```
int* p = new int{3};

int* q = new int[]{ 3, 4, 5 };
```

The heap

```
int* p = new int{3};

int* q = new int[]{ 3, 4, 5 };

int* r = new int[32];
```

The heap

```
int* p = new int{3};

int* q = new int[]{ 3, 4, 5 };

int* r = new int[32];

int* s = new int[w * h];
```

The heap

```
int* p = new int{3};                delete p;

int* q = new int[]{ 3, 4, 5 };

int* r = new int[32];

int* s = new int[w * h];
```

The heap

```
int* p = new int{3};                delete p;

int* q = new int[]{ 3, 4, 5 };      delete [] q;

int* r = new int[32];               delete [] r;

int* s = new int[w * h];            delete [] s;
```

A rudimentary vector

```
struct Int_vector
{
    size_t size;
    size_t capacity;
    int* data;
};
```