# Pointers for Arrays

1/24/25

# Administrivia

- HW 3 (memory diagrams and binary) due Monday night

# Recall: Pointers

- Variable to store an address in memory
  - Includes variable type

```
int* ptr = &x;
*ptr = 23;
```

# Another mystery function problem

# Dynamic memory allocation

- Can’t do with stack-allocated memory:
  - Return pointer to it
  - Get variable-sized part of it (sometimes actually possible)

# Dynamic memory allocation

- Can't do with stack-allocated memory:
  - Return pointer to it
  - Get variable-sized part of it (sometimes actually possible)

- Need memory allocated from the heap:

```
int* ptr = (int*) malloc(sizeof(int));
```

cast; malloc returns void*

gives number of bytes in an int

# Other notes on malloc

- Need to #include <stdlib.h>

- Need to explicitly free dynamically-allocated memory
  - pass pointer once to free:

```
free(ptr);
```

# Pointers as arrays

- Just like with new in Java, malloc can allocate space for more than one data item:

```
int* a = (int*) malloc(5*sizeof(int));
```

- Then can access the items with array syntax:

```
a[2] = a[1];
```

# Strings in C

- Just an array of chars with a 0 at the end (not ‘0’)
  - Type is a char*
- For “methods”, use functions from string.h
- Things to remember:
  - Allocate room for the \0
  - Scanf (with %s) takes a char*, not a char**

# Which of the following lines does not do as its comment says?

```
#include <stdio.h>                  //for malloc  (line 1)
...
int* p = (int*) malloc(10);         //allocate 10 ints  (line 2)
char* s = (char*) malloc(6);        //allocate room for “David” (line 3)
```

A. Line 1 does not match its comment
B. Line 2 does not match its comment
C. Line 3 does not match its comment
D. More than 1 line does not match its comment
E. All lines match their comments

# Which of the following lines does not do as its comment says?

```
#include <stdio.h>                    //for malloc  (line 1)
...
int* p = (int*) malloc(10);           //allocate 10 ints  (line 2)
char* s = (char*) malloc(6);          //allocate room for "David" (line 3)
```

A. Line 1 does not match its comment
B. Line 2 does not match its comment
C. Line 3 does not match its comment
D. More than 1 line does not match its comment (1, 2, and mostly 3)
E. All lines match their comments

# Which of the following is true for the code below?

```
int nums[10];
char* s = (char*) malloc(100 * sizeof(char));
char* t = s;
```

A. free should not be called on nums
B. free should not be called on s
C. free should not be called on t
D. More than one statement above is true
E. None of the statements above are true

# Which of the following is true for the code below?

```
int nums[10];
char* s = (char*) malloc(100 * sizeof(char));
char* t = s;
```

A. free should not be called on nums
B. free should not be called on s
C. free should not be called on t
D. More than one statement above is true
E. None of the statements above are true

(free can be called on either s or t, but not both)

# Which of the following is a problem with the code below?

```
int* f(int x, double* y) {
    int z = 23;
    if(x < *y)
        z = (int) (z - *y);
    return &z;
}
```

A. Type error involving argument y
B. Type mismatch between signature and return value
C. Shouldn't return a pointer to a local variable
D. Other syntax error
E. The code is fine (albeit not useful)

# Which of the following is a problem with the code below?

```
int* f(int x, double* y) {
    int z = 23;
    if(x < *y)
        z = (int) (z - *y);
    return &z;
}
```

A. Type error involving argument y
B. Type mismatch between signature and return value
C. Shouldn't return a pointer to a local variable
D. Other syntax error
E. The code is fine (albeit not useful)

# Which of the following lines has an error?

```
int x = 3;
char[10] str;                  //A
char* p = str;                 //B

scanf("%d", &x);               //C
scanf("%s", &str);             //D
                               //E: Not exactly one
                               //     of the above
```

# Which of the following lines has an error?

```
int x = 3;
char[10] str;                  //A
char* p = str;                 //B

scanf("%d", &x);               //C
scanf("%s", &str);             //D
                               //E: Not exactly one
                               //    of the above (A&D)
```