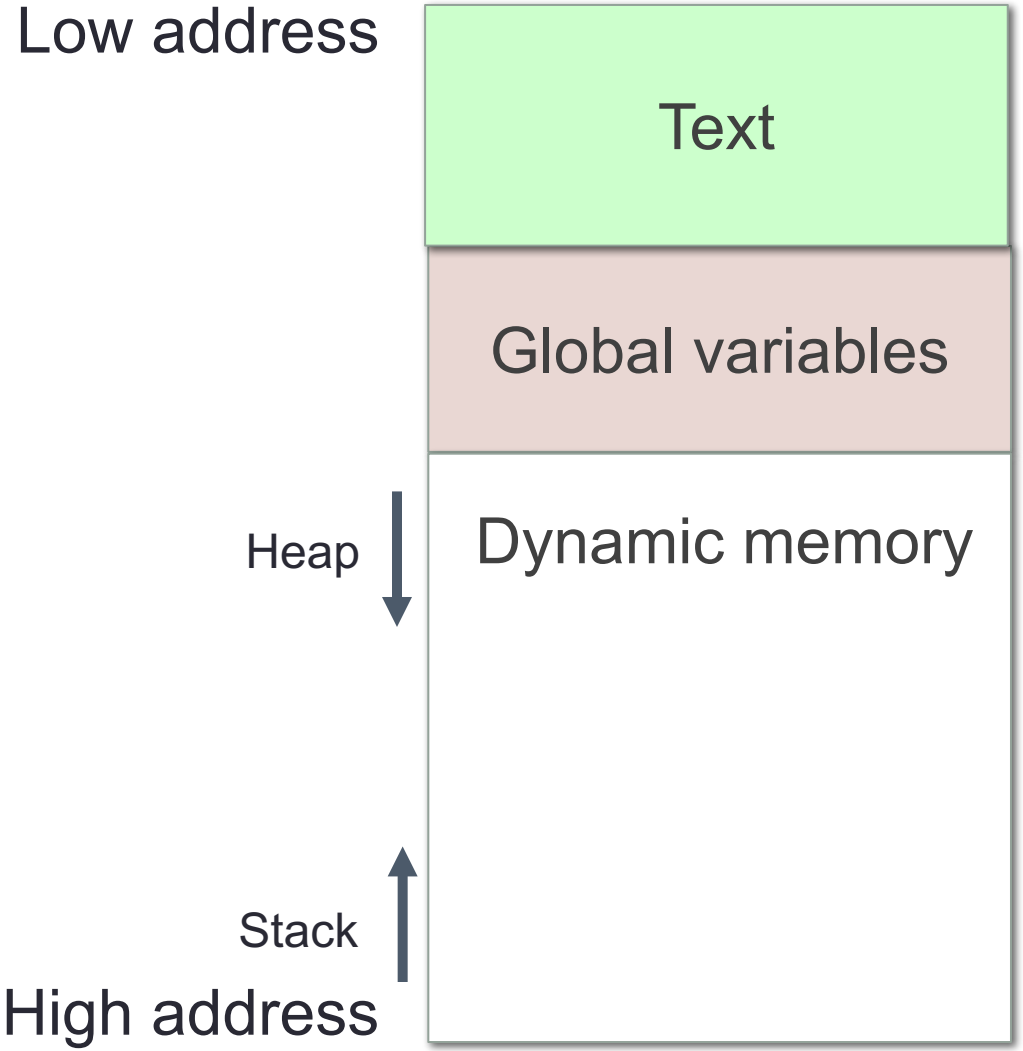
DYNAMIC MEMORY ALLOCATION LINKED LISTS

Problem Solving with Computers-I

Review: Structs, arrays of structs

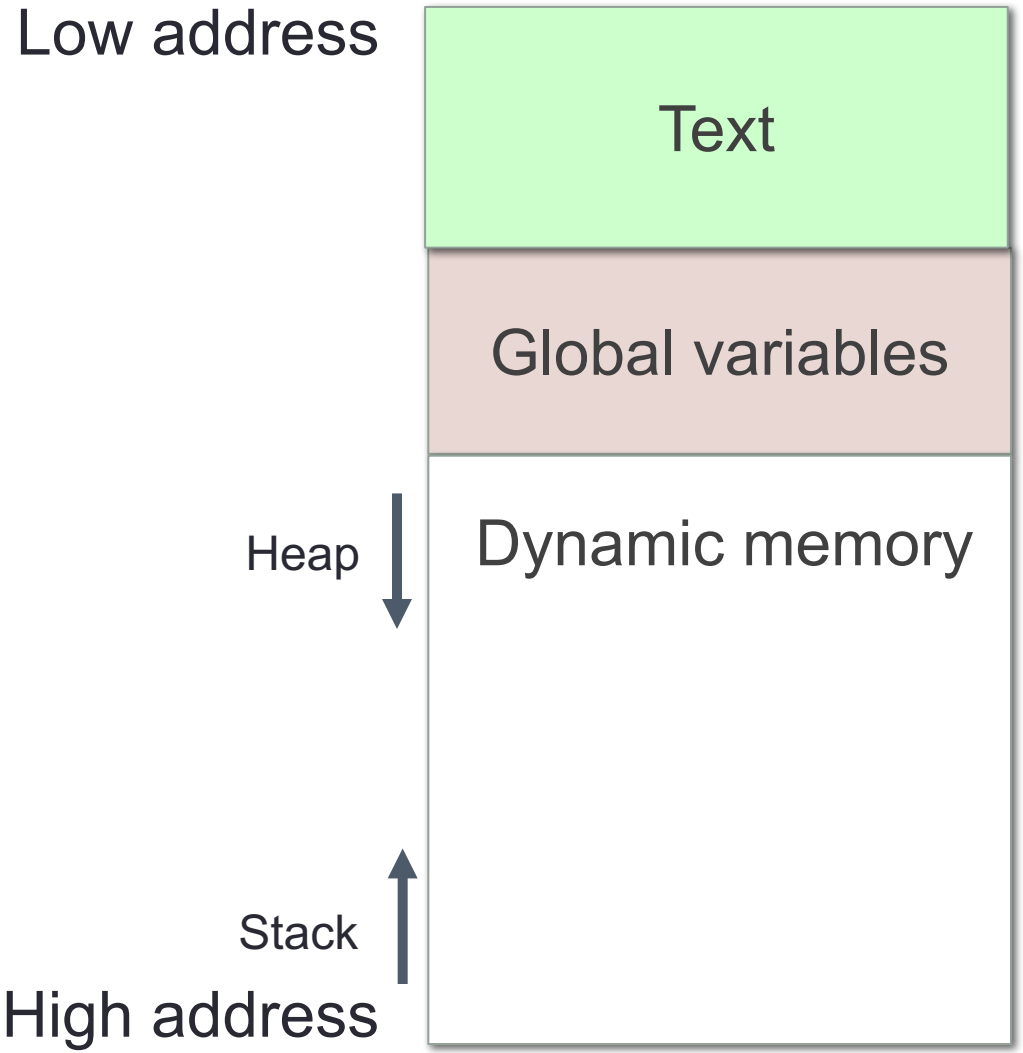
Program layout in memory at runtime

A generic model for memory



Creating data on the heap: new and delete

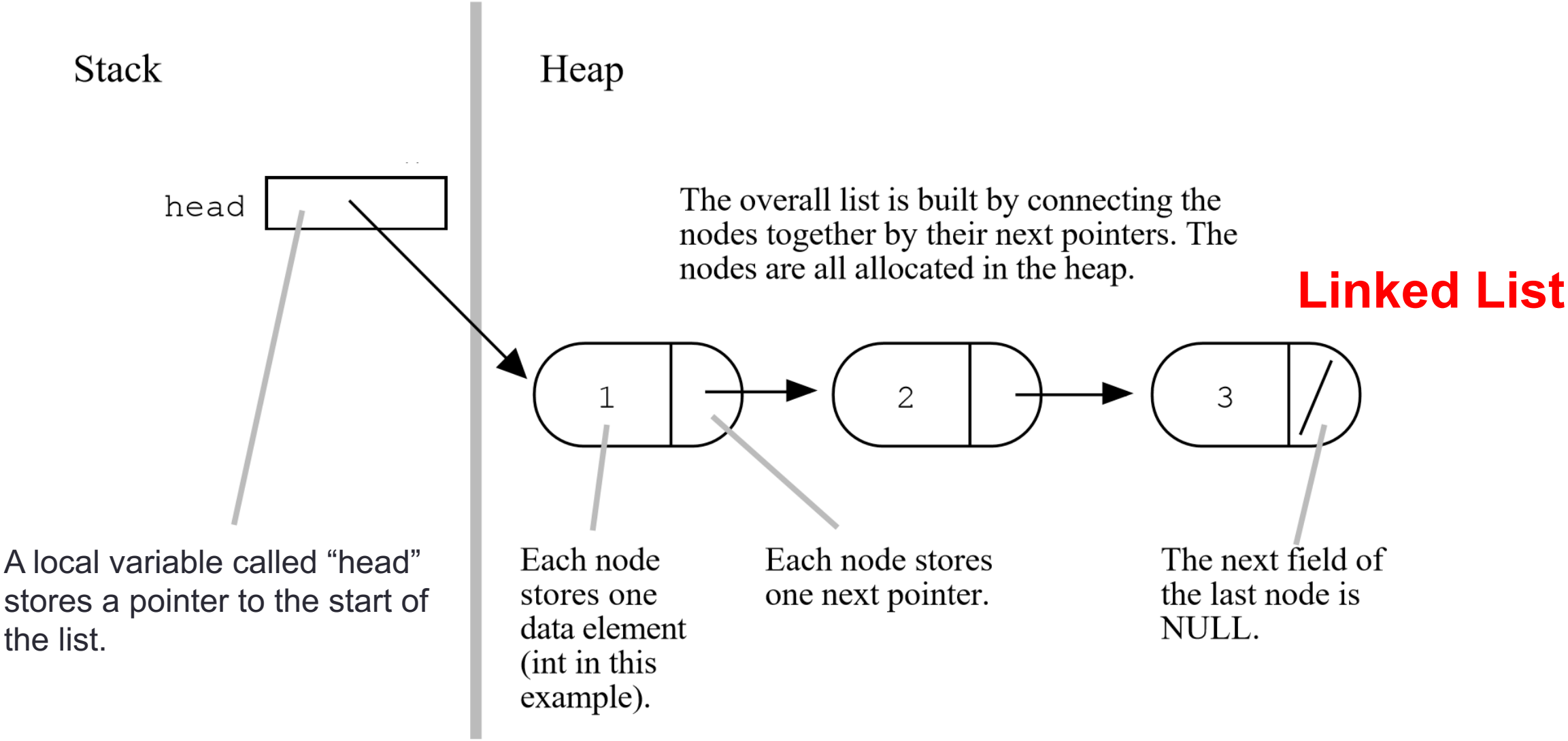
```
void foo() {
  int *n = NULL;
  n = new int;
  *n = 10;

  int *arr = new int[5];
  arr[0] = arr[1] = … = arr[4] = 42;

  delete n;
  delete[] arr;
}
```



Linked Lists

The Drawing Of List {1, 2, 3}

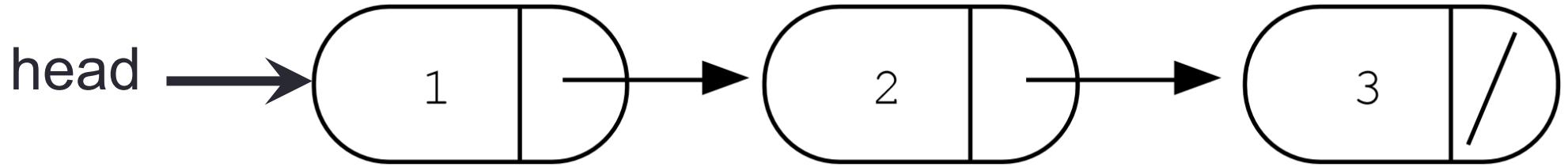
Array List



Linked List

Accessing elements of a list

```
struct Node {
      int data;
      Node *next;
};
```



Assume the linked list has already been created, what do the following expressions evaluate to?

1. head->data
2. head->next->data
3. head->next->next->data
4. head->next->next->next->data

A. 1
B. 2
C. 3
D. NULL
E. Run time error

Creating a small list

- Define an empty list
- Add a node to the list with data = 10

```
struct Node {
      int data;
      Node *next;
};
```

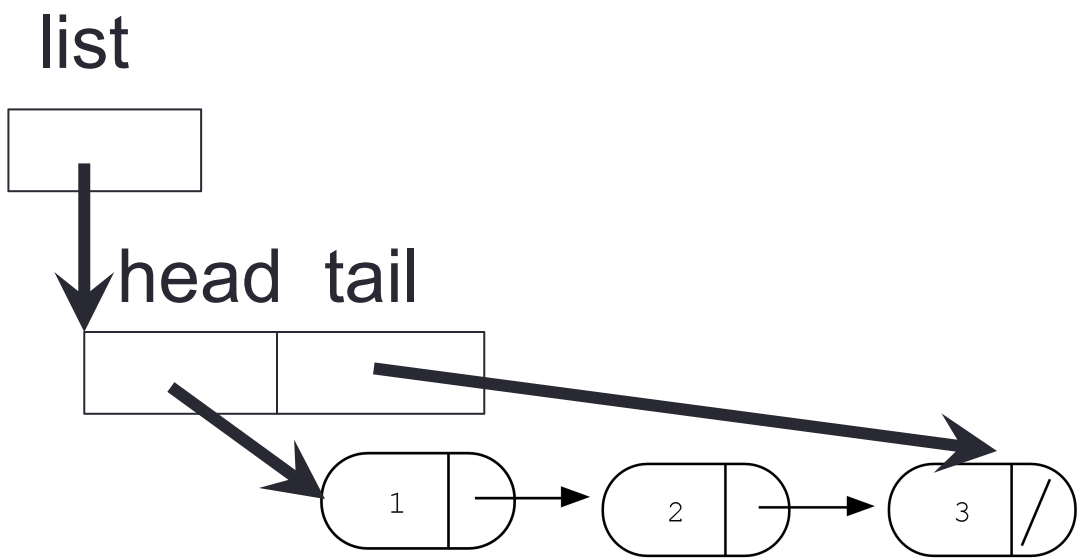
Building a list from an array

```
LinkedList* arrayToLinkedList(int a[], int size);
```

a

Iterating through the list

```
int lengthOfList(LinkedList * list) {
   /* Find the number of elements in the list */




}
```



Deleting the list

```
int freeLinkedList(LinkedList * list) {
   /* Free all the memory that was created on the heap */




}
```