C++ MEMORY MODEL LINKED LISTS

Problem Solving with Computers-I

The case of the disappearing data!

```
int getInt(){
      int x=5;
      return x;
}
int* getAddressOfInt(){
      int x=10;
      return &x;
}
int main(){
      int y=0, *p=nullptr, z=0;
      y = getInt();
      p = getAddressOfInt();
      z = *p;
     cout<<y<<", "<<z<<", "<<*p<<endl;
}
```

What is the output?

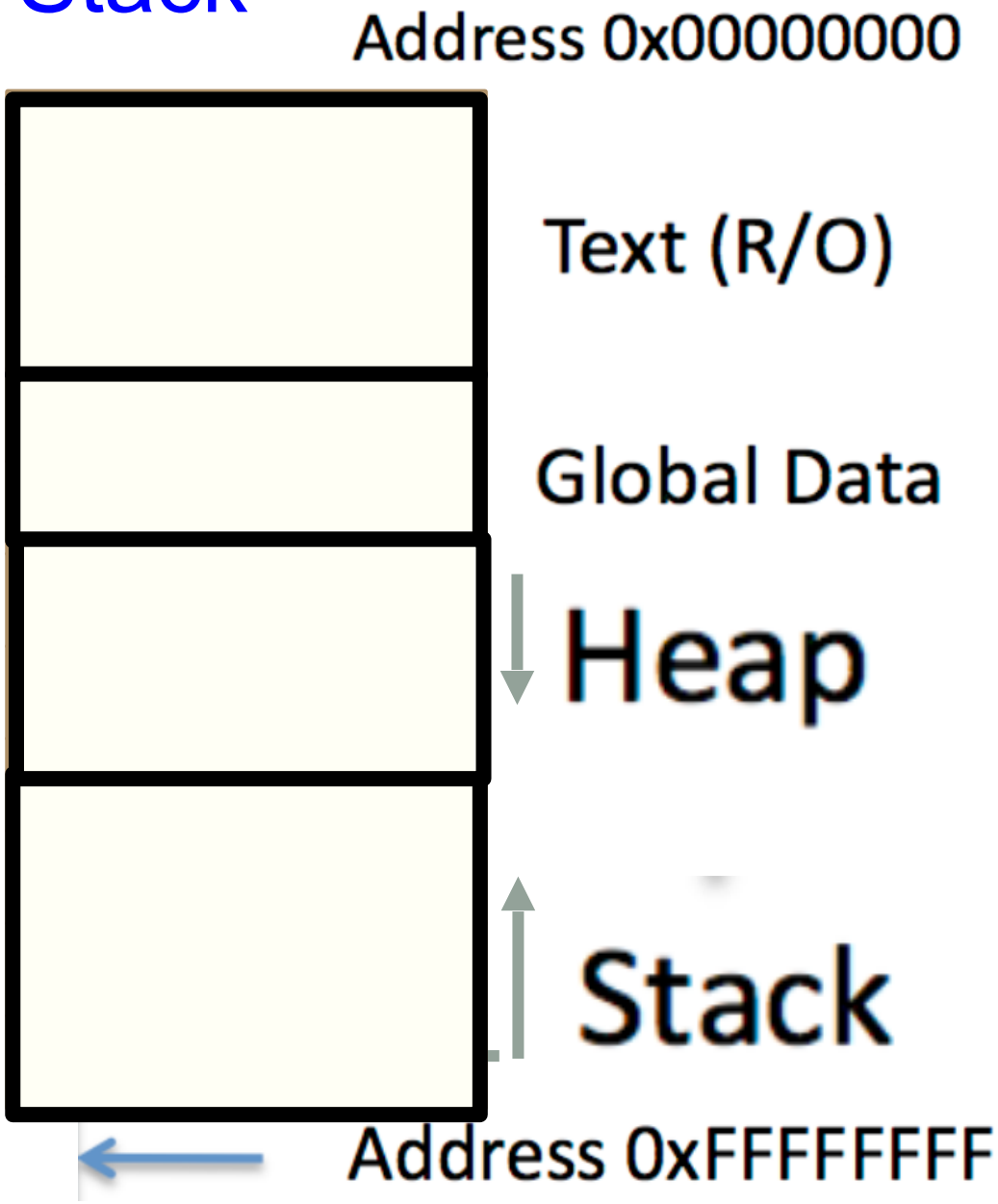
A. 5, 0, 10

B. 5, 10, 10

C. Something else

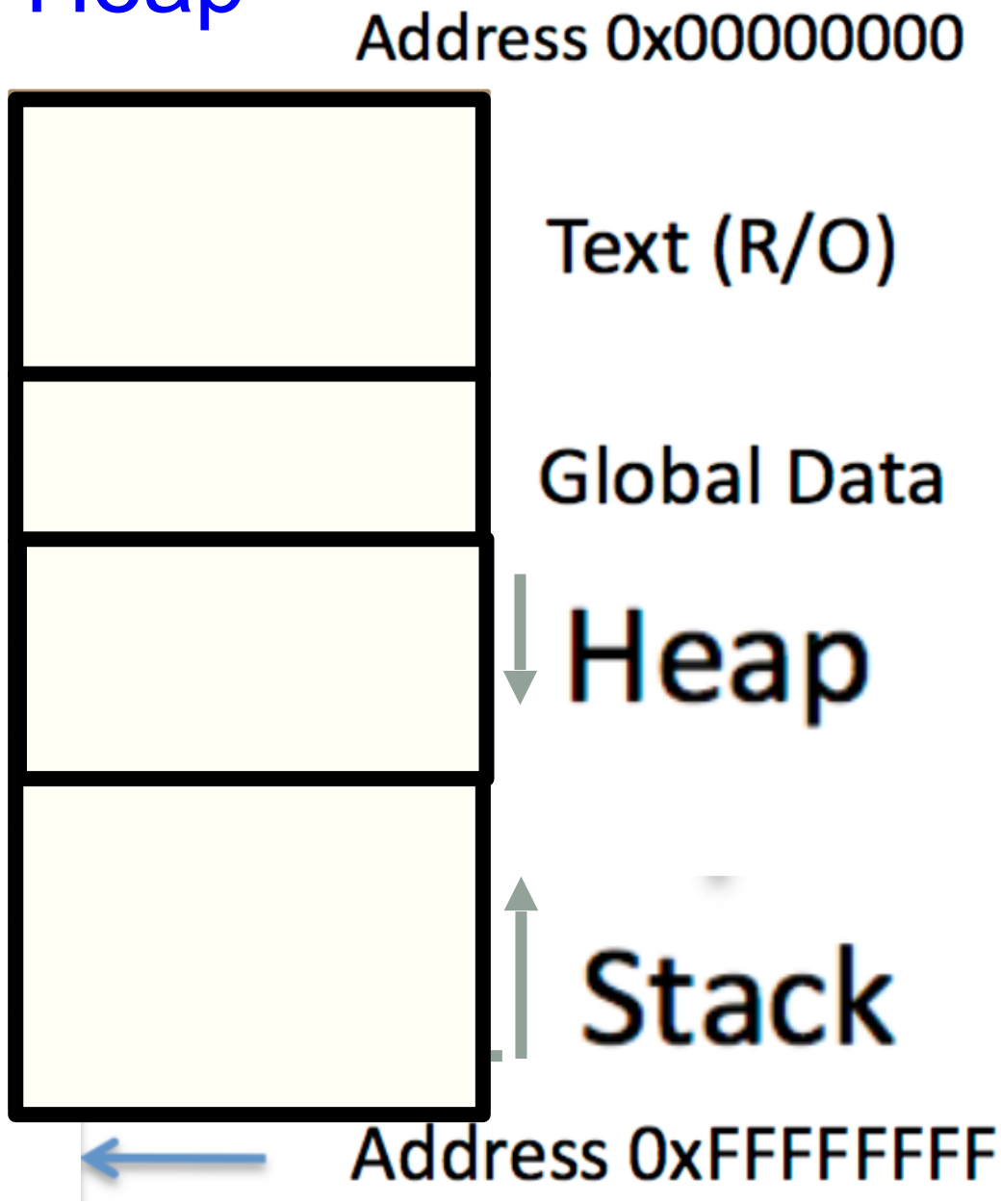
C++ Memory Model: Stack

- Stack: Segment of memory managed automatically using a Last in First Out (LIFO) principle
- Think of it like a stack of books!



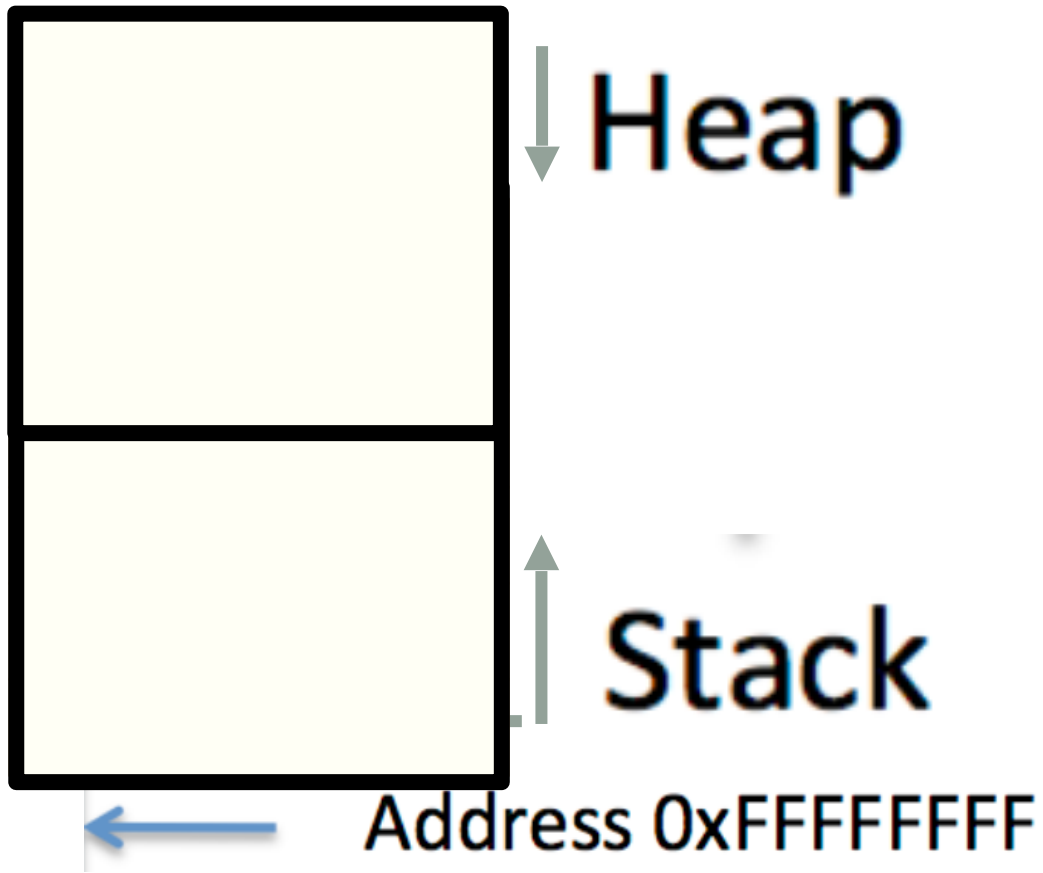
C++ Memory Model: Heap

- Heap: Segment of memory managed by the programmer
- Data created on the heap stays there
  - FOREVER or
  - until the programmer explicitly deletes it



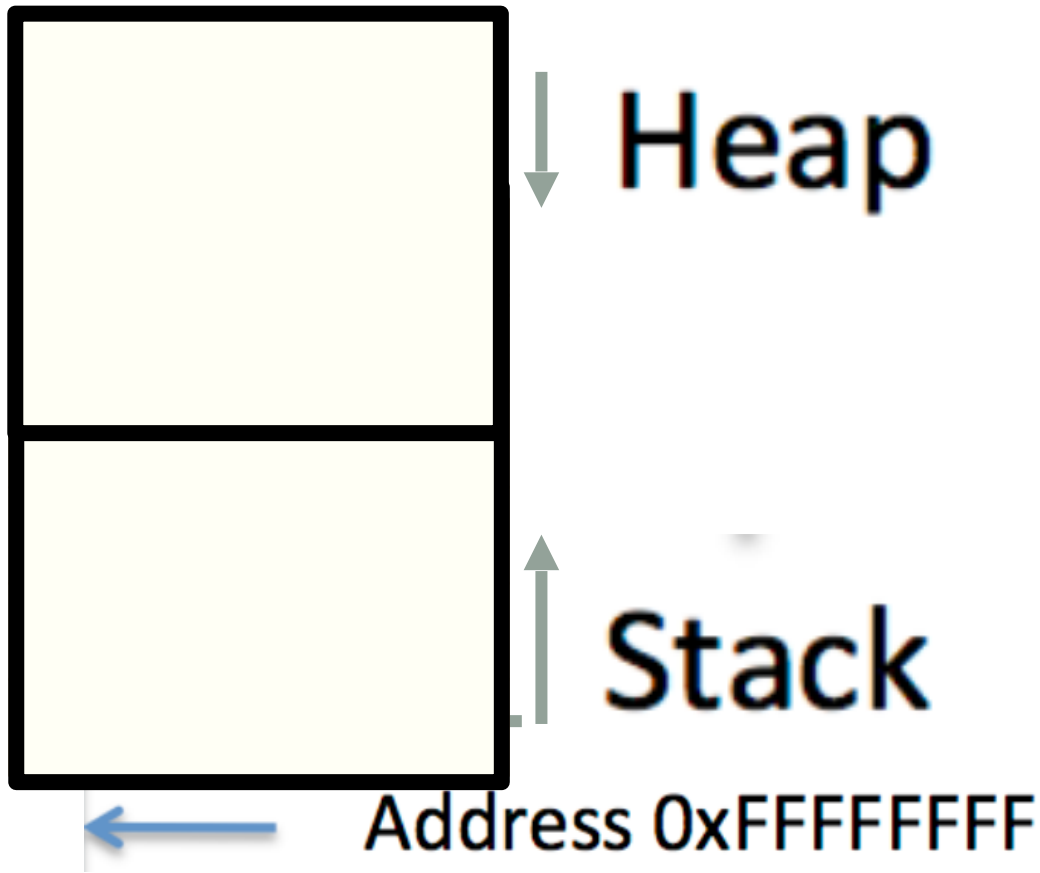
Creating data on the Heap: new

To **allocate** memory on the heap use the **new** operator



Deleting data on the Heap: delete

To **free** memory on the heap use the **delete** operator



Dynamic memory management = Managing data on the heap

```
int *p= new int; //create an integer on the heap

SuperHero *n = new SuperHero;

            //create a Student on the heap

delete p; //Frees the integer

delete n; //Frees the Student
```

Heap vs. stack

```
1 #include <iostream>
2 using namespace std;
3
4 int* createAnIntArray(int len){
5
6     int arr[len];
7     return arr;
8
9 }
```

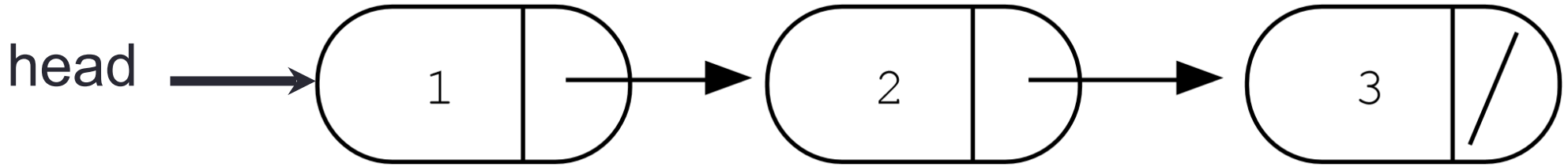
Does the above function correctly return an array of integers?

A. Yes

B. No

Accessing elements of a linked 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;
};
```

Heap vs. stack

```
// Post-condition: create a two-node linked list
// and return the address of the head of the linked list
Node* createSmallLinkedList(int x, int y){
    Node *head = NULL;
    Node n1 ={x, NULL};
    Node n2 ={y, NULL};
    head = &n1;
    n1.next = &n2;
    return head;
}
```

Is the above function correct?

A. Yes

B. No

Creating a small list

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

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

struct LinkedList {
      Node *head;
      Node *tail;
};
```

Inserting a node in a linked list

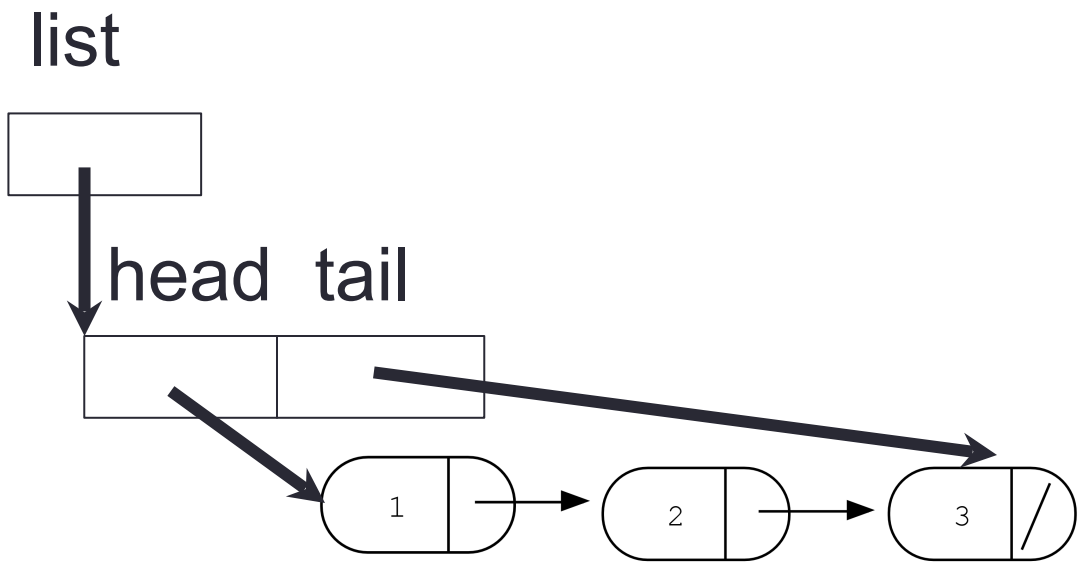
```
void insert(LinkedList *h, int value) ;
```

Iterating through the list

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




}
```



Next time

- Memory-related errors
- Double-linked lists