LINKED LISTS & DYNAMIC MEMORY ERRORS

Problem Solving with Computers-I

Creating a small list

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

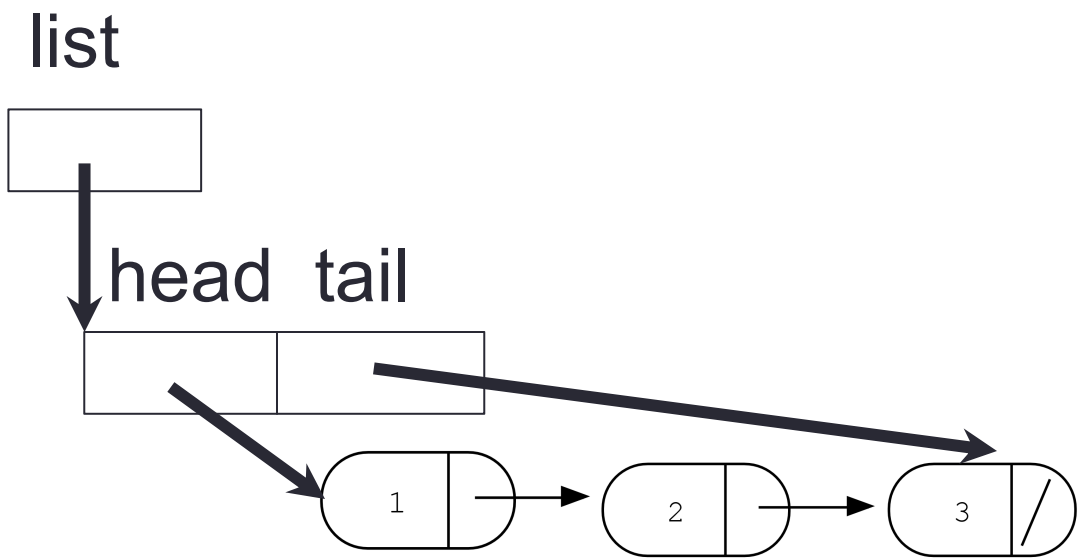
```
struct Node {
      int data;
      Node* next;
};

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

Inserting a node at the head of a linked list

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

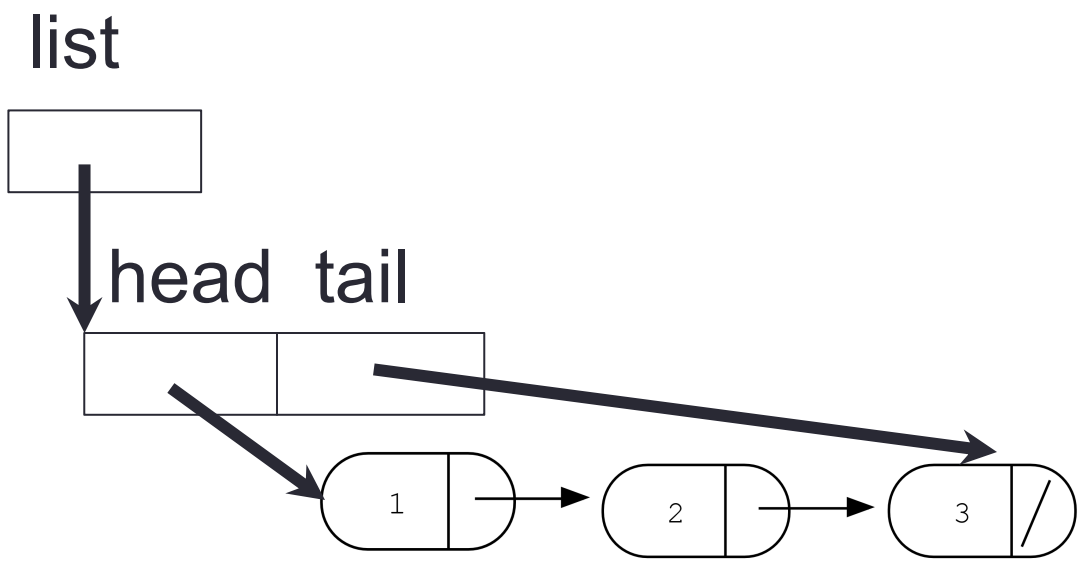
Iterating through the list

```
/* Find the number of elements in the list */

int count(LinkedList& list);
```



Deleting the list

```
/* Free all the memory that was created on the heap*/

void freeList(LinkedList& list);
```



Deleting a node from the linked list

Double Linked Lists

Array List

Implementing a double-linked list

- Define a node in a double linked list
- Write functions to
  - insert a node to the head/tail of the linked list
  - Print all the elements of the list
  - Delete a node with a given value
  - Free the list

Dangling pointers and memory leaks

–Dangling pointer: Pointer points to a memory location that no longer exists

–Memory leaks (tardy free):

- Heap memory not deallocated before the end of program
- Heap memory that can no longer be accessed

Dynamic memory pitfalls

Memory leaks (tardy free):

Heap memory not deallocated before the end of program

Heap memory that can no longer be accessed

Example

```
void foo(){
     int* p = new int;

}
```

Memory errors can cause your program to crash

- **Segmentation faults**: Program crashes because it attempted to access a memory location that either doesn't exist or doesn't have permission to access
- Examples of code that results in undefined behavior and potential segmentation fault

```
int arr[] = {50, 60, 70};

for(int i=0; i<=3; i++){
   cout<<arr[i]<<endl;
}
```

```
int x = 10;
int* p;
cout<<*p<<endl;
```

Detecting memory errors

- Valgrind is a tool that reports errors related to dynamic memory allocation, access and deletion
- Run valgrind on your program using the following command:

```
valgrind --leak-check=full ./myprog
```

Next time

- Recursion