

INTRO TO PA01

OPERATOR OVERLOADING

RECURSION

GDB

Problem Solving with Computers-II

C++

```
#include <iostream>
using namespace std;

int main(){
    cout<<"Hola Facebook\n";
    return 0;
}
```



Announcements

- PA01 released, due in one week
- Midterm next week (Thurs)(08/29) - All topics covered until Tuesday of next week (Linked Lists and BST).

For more details visit <https://ucsb-cs24.github.io/m19/exam/e01/>

PA01: Card matching game with linked lists

Alice:

<p>3♥</p>  <p>Genevieve Bell Australian National Univ. Director - Autonomy, Agency and Assurance Institute, ABI Woman of Vision, WITI Hall of Fame. <i>Known for:</i> combining anthropology and tech to explore social, cultural aspects of ubiquitous computing. http://en.wikipedia.org/wiki/Genevieve_Bell</p> <p>♥ 3</p>	<p>2♠</p>  <p>Fran Bilas ENIAC computer programmer team 1946, WITI Hall of Fame. <i>Known for:</i> being a pioneer in programming the first electronic general-purpose computer. http://en.wikipedia.org/wiki/Fran_Bilas</p> <p>♠ 2</p>	<p>A♣</p>  <p>Vicki Hanson CEO of ACM, Former RIT Distinguished Prof., Prof. Univ. of Dundee, Fellow Royal Society of Edinburgh, ACM Fellow, ABI Woman of Vision. <i>Known for:</i> contributions to computing technologies for people with disabilities. http://en.wikipedia.org/wiki/Vicki_L._Hanson</p> <p>♣ A</p>	<p>3♣</p>  <p>Sophie Wilson Designer Acorn Microcomputer, Broadcom Director IC Design, Computer History Museum Fellow, Fellow of the Royal Society. <i>Known for:</i> computer hardware design and for leadership in the transgender technical community. http://en.wikipedia.org/wiki/Sophie_Wilson</p> <p>♣ 3</p>	<p>9♥</p>  <p>Irene Greif ABIE Award for Technical Leadership, IBM User Experience Group, ACM Fellow, AAAS Fellow, Formed Lotus Research 1992. <i>Known for:</i> pioneering the field of Computer Supported Cooperative Work. http://en.wikipedia.org/wiki/Irene_Greif</p> <p>♥ 9</p>	<p>A♠</p>  <p>Radia Perlman Intel Fellow, IEEE and ACM Fellow, first ABI Woman of Vision award winner, National Inventors Hall of Fame, Internet Hall of Fame. <i>Known for:</i> contributions to network routing and security protocols. http://en.wikipedia.org/wiki/Radia_Permalink</p> <p>♠ A</p>
---	---	--	--	--	--

Bob:

<p>2♣</p>  <p>Jean Bartik ENIAC computer programmer team 1946, Fellow Computer History Museum, IEEE Computer Pioneer Award. <i>Known for:</i> being a pioneer in programming the first electronic general-purpose computer. http://en.wikipedia.org/wiki/Jean_Bartik</p> <p>♣ 2</p>	<p>A♠</p>  <p>Radia Perlman Intel Fellow, IEEE and ACM Fellow, first ABI Woman of Vision award winner, National Inventors Hall of Fame, Internet Hall of Fame. <i>Known for:</i> contributions to network routing and security protocols. http://en.wikipedia.org/wiki/Radia_Permalink</p> <p>♠ A</p>	<p>J♦</p>  <p>Yuying Gao Former IBM Distinguished Engineer, ABI Women of Vision, IEEE Fellow. <i>Known for:</i> contributions to speech recognition and speech-to-speech translation. https://en.wikipedia.org/wiki/Yuying_Gao</p> <p>♦ J</p>	<p>9♥</p>  <p>Irene Greif ABIE Award for Technical Leadership, IBM User Experience Group, ACM Fellow, AAAS Fellow, Formed Lotus Research 1992. <i>Known for:</i> pioneering the field of Computer Supported Cooperative Work. http://en.wikipedia.org/wiki/Irene_Greif</p> <p>♥ 9</p>	<p>3♣</p>  <p>Sophie Wilson Designer Acorn Microcomputer, Broadcom Director IC Design, Computer History Museum Fellow, Fellow of the Royal Society. <i>Known for:</i> computer hardware design and for leadership in the transgender technical community. http://en.wikipedia.org/wiki/Sophie_Wilson</p> <p>♣ 3</p>
---	---	---	---	---

Review PA01: Card matching game with linked lists

Correct output after running `make && ./game alice_cards.txt bob_cards.txt`:

Alice picked matching card c 3
 Bob picked matching card s a
 Alice picked matching card h 9

Alice's cards:

h 3
 s 2
 c a

Bob's cards:

c 2
 d j

Note: 0=10, a=ace, k=king, q=queen, j=jack

Contents of `alice_cards.txt`:



Contents of `bob_cards.txt`:



Overloading Binary Comparison Operators

We would like to be able to compare two objects of the class using the following operators

`==`

`!=`

and possibly others

Last class: overloaded `==` for `LinkedList`

Overloading input/output stream

Wouldn't it be convenient if we could do this:

```
LinkedList list;  
cout<<list; //prints all the elements of list
```

Overloading Binary Arithmetic Operators

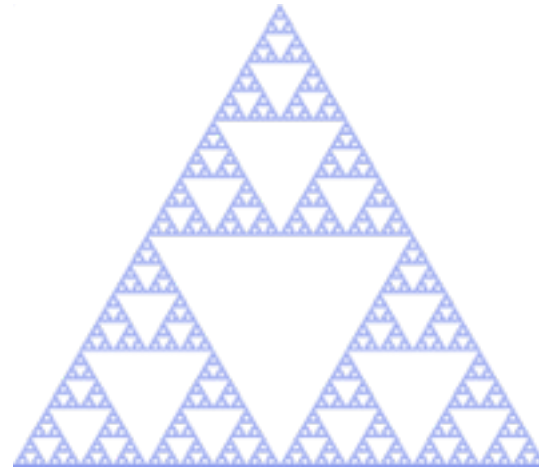
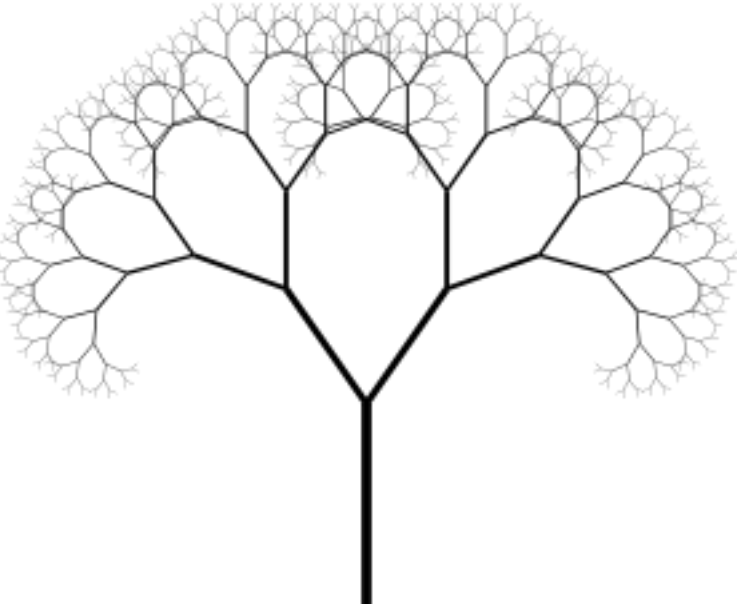
We would like to be able to add two points as follows

```
LinkedList l1, l2;
```

```
//append nodes to l1 and l2;
```

```
LinkedList l3 = l1 + l2 ;
```

Recursion



Sierpinski triangle



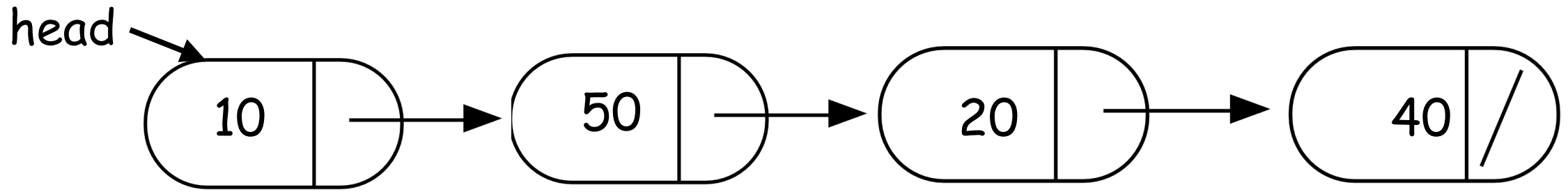
Zooming into a Koch's snowflake



Describe a linked-list recursively

Common methods of linked list that can be implemented using recursion

- Sum all the values
- Print all the values
- Search for a value
- Delete all the nodes in a linked list



```
int IntList::sum() {
```

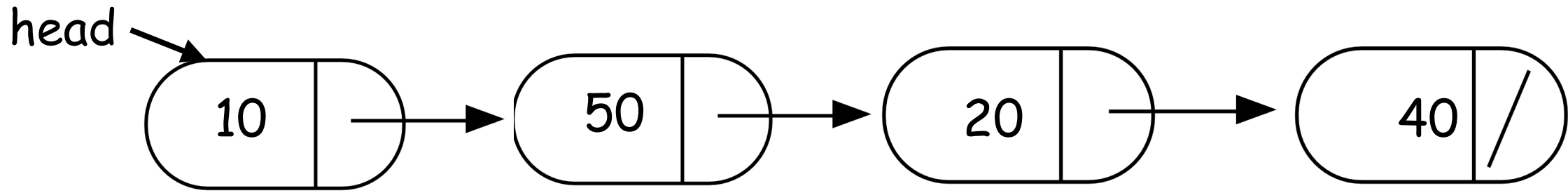
```
    //Return the sum of all elements in a linked list  
}
```

Helper functions

- Sometimes your functions takes an input that is not easy to recurse on
- In that case define a new function with appropriate parameters: This is your helper function
- Call the helper function to perform the recursion
- Usually the helper function is private

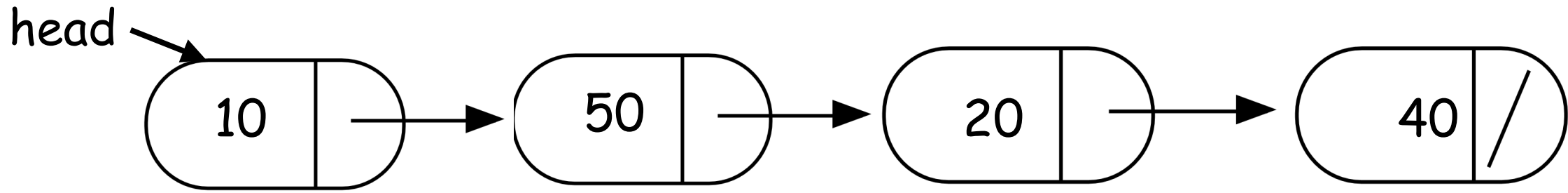
For example

```
Int IntList::sum( ) {  
  
    return sum(head) ;  
    //helper function that performs the recursion.  
  
}
```



```
int IntList::sum(Node* p) {
```

```
}
```



```
bool IntList::clear(Node* p) {
```

```
}
```

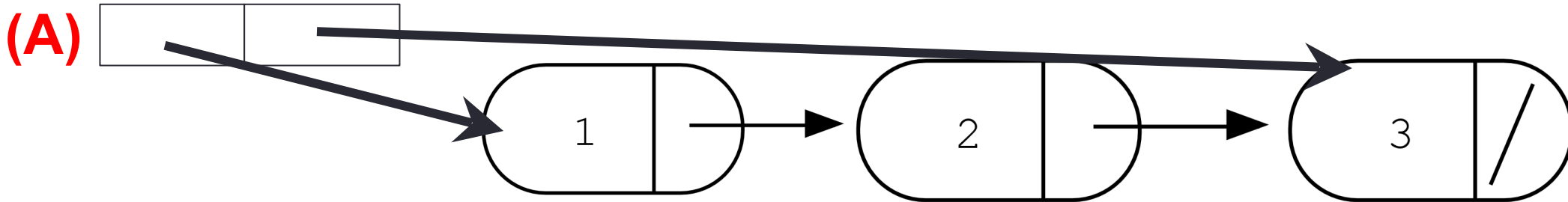

Concept Question

```
LinkedList::~~LinkedList(){  
    delete head;  
}
```

```
class Node {  
    public:  
        int info;  
        Node *next;  
};
```

Which of the following objects are deleted when the destructor of Linked-list is called?

head tail



(B): only the first node

(C): A and B

(D): All the nodes of the linked list

(E): A and D

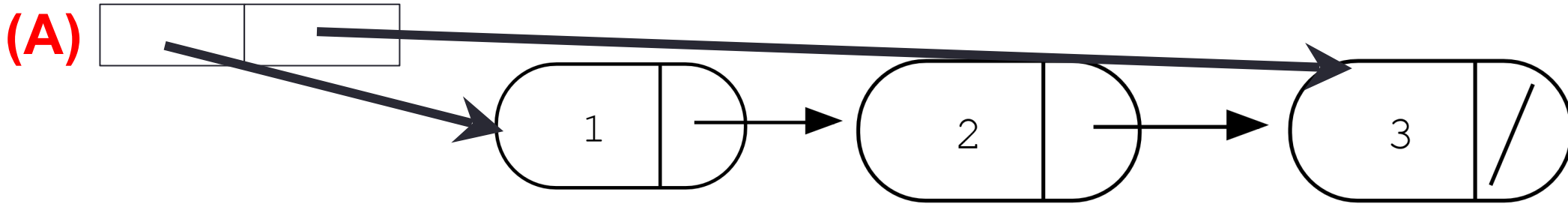
Concept question

```
LinkedList::~~LinkedList(){  
    delete head;  
}
```

```
Node::~~Node(){  
    delete next;  
}
```

Which of the following objects are deleted when the destructor of Linked-list is called?

head tail



(B): All the nodes in the linked-list

(C): A and B

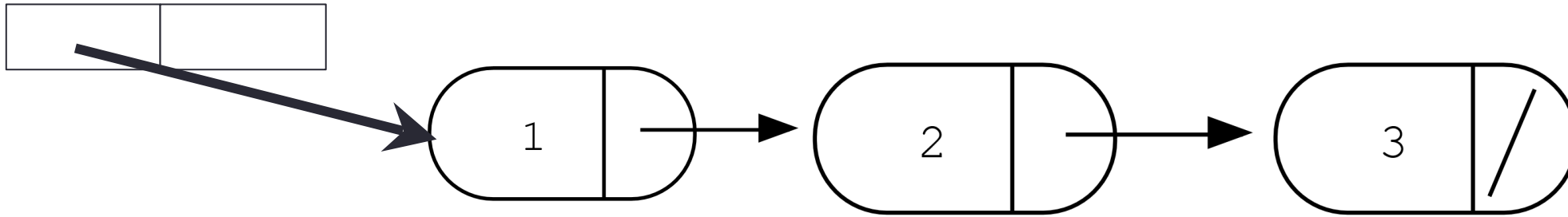
(D): Program crashes with a segmentation fault

(E): None of the above

```
LinkedList::~~LinkedList(){  
    delete head;  
}
```

```
Node::~~Node(){  
    delete next;  
}
```

head tail



GDB: GNU Debugger

- To use gdb, compile with the -g flag
- Setting breakpoints (b)
- Running programs that take arguments within gdb (r arguments)
- Continue execution until breakpoint is reached (c)
- Stepping into functions with step (s)
- Stepping over functions with next (n)
- Re-running a program (r)
- Examining local variables (info locals)
- Printing the value of variables with print (p)
- Quitting gdb (q)
- Debugging segfaults with backtrace (bt)

* Refer to the gdb cheat sheet: <https://ucsb-cs24.github.io/m19/lectures/GDB-cheatsheet.pdf>

Next time

- Binary Search Trees