

# OPERATOR OVERLOADING

## GDB

## RECURSION

## INTRO TO PA01

---

Problem Solving with Computers-II

# C++

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

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


GitHub



# PA01: Card matching game with linked lists

Alice:

3♥




**Genevieve Bell**  
Australian National Univ.  
Director - Autonomy, Agency and Awareness Institute, ABI Woman of Vision, WITI Hall of Fame.  
*Known for* combining anthropology and tech to explore social, cultural aspects of ubiquitous computing.

[http://en.wikipedia.org/wiki/Genevieve\\_Bell](http://en.wikipedia.org/wiki/Genevieve_Bell)

♠♣

2♠



**Fran Bilas**  
ENIAC computer programmer team 1946, WITI Hall of Fame.  
*Known for* being a pioneer in programming the first electronic general-purpose computer.

[http://en.wikipedia.org/wiki/Fran\\_Bilas](http://en.wikipedia.org/wiki/Fran_Bilas)

♥♦

A♣



**Vicki Hanson**  
CEO of ACM, former RIT Distinguished Prof., Prof. Univ. of Dundee, Fellow Royal Society of Edinburgh, ACM Fellow, ABI Woman of Vision.  
*Known for* contributions to computing technologies for people with disabilities.

[http://en.wikipedia.org/wiki/Vicki\\_L.\\_Hanson](http://en.wikipedia.org/wiki/Vicki_L._Hanson)

♠♣

3♣




**Sophie Wilson**  
Designer Acorn Microcomputer, Broadcom Director IC Design, Computer History Museum Fellow, Fellow of the Royal Society.  
*Known for* computer hardware design and for leadership in the transgender technical community.

[http://en.wikipedia.org/wiki/Sophie\\_Wilson](http://en.wikipedia.org/wiki/Sophie_Wilson)

♠♣

9♥



**Irene Greif**  
ABIE Award for Technical Leadership, IBM User Experience Group, ACM Fellow, AAAS Fellow, Formed Lotus Research 1992.  
*Known for* pioneering the field of Computer Supported Cooperative Work.

[http://en.wikipedia.org/wiki/Irene\\_Greif](http://en.wikipedia.org/wiki/Irene_Greif)

♥♦

A♠




**Radia Perlman**  
Intel Fellow, IEEE and ACM Fellow, first ABI Woman of Vision award winner, National Inventors Hall of Fame, Internet Hall of Fame.  
*Known for* contributions to network routing and security protocols.

[http://en.wikipedia.org/wiki/Radia\\_Perlman](http://en.wikipedia.org/wiki/Radia_Perlman)

♥♠

Bob:

2♣




**Jean Bartik**  
ENIAC computer programmer team 1946, Fellow Computer History Museum, IEEE Computer Pioneer Award.  
*Known for* being a pioneer in programming the first electronic general-purpose computer.

[http://en.wikipedia.org/wiki/Jean\\_Bartik](http://en.wikipedia.org/wiki/Jean_Bartik)

♠♣

A♠




**Radia Perlman**  
Intel Fellow, IEEE and ACM Fellow, first ABI Woman of Vision award winner, National Inventors Hall of Fame, Internet Hall of Fame.  
*Known for* contributions to network routing and security protocols.

[http://en.wikipedia.org/wiki/Radia\\_Perlman](http://en.wikipedia.org/wiki/Radia_Perlman)

♥♠

J♦



**Yuqing Gao**  
Former IBM Distinguished Engineer, ABI Women of Vision, IEEE Fellow.  
*Known for* contributions to speech recognition and speed-to-speech translation.

[http://en.wikipedia.org/wiki/Yuqing\\_Gao](http://en.wikipedia.org/wiki/Yuqing_Gao)

♥♦

9♥



**Irene Greif**  
ABIE Award for Technical Leadership, IBM User Experience Group, ACM Fellow, AAAS Fellow, Formed Lotus Research 1992.  
*Known for* pioneering the field of Computer Supported Cooperative Work.

[http://en.wikipedia.org/wiki/Irene\\_Greif](http://en.wikipedia.org/wiki/Irene_Greif)

♥♦

3♣



**Sophie Wilson**  
Designer Acorn Microcomputer, Broadcom Director IC Design, Computer History Museum Fellow, Fellow of the Royal Society.  
*Known for* computer hardware design and for leadership in the transgender technical community.

[http://en.wikipedia.org/wiki/Sophie\\_Wilson](http://en.wikipedia.org/wiki/Sophie_Wilson)

♠♣

# 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

```

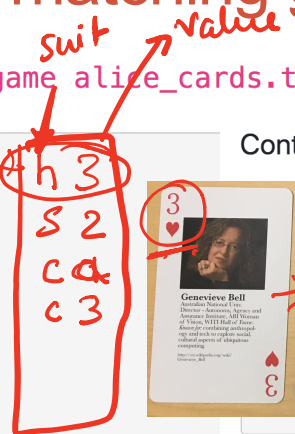
Bob's cards:

```

c 2
d j

```

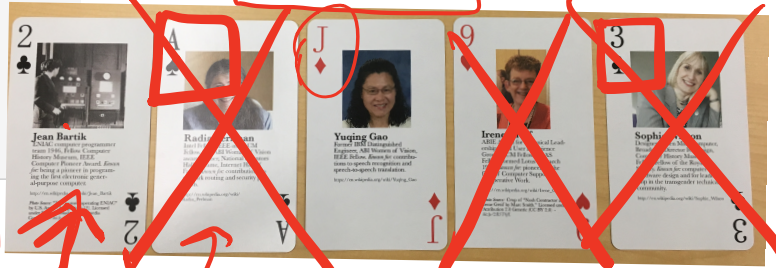
Note: 0=10, a=ace, k=king, q=queen, j=jack



Contents of `alice_cards.txt`:



Contents of `bob_cards.txt`:



# 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: <http://darkdust.net/files/GDB%20Cheat%20Sheet.pdf>

# 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

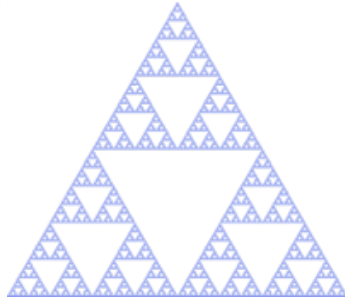
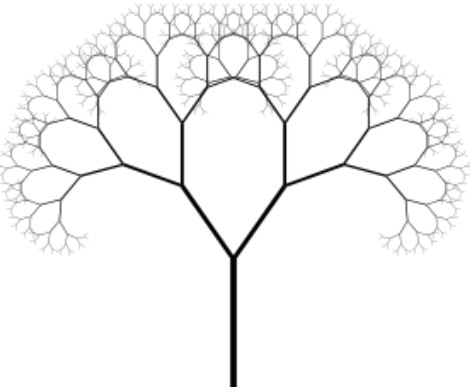
We would like to be able to add two points as follows

```
LinkedList l1, l2;
```

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

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

# Recursion



Sierpinski triangle



Zooming into a Koch's snowflake

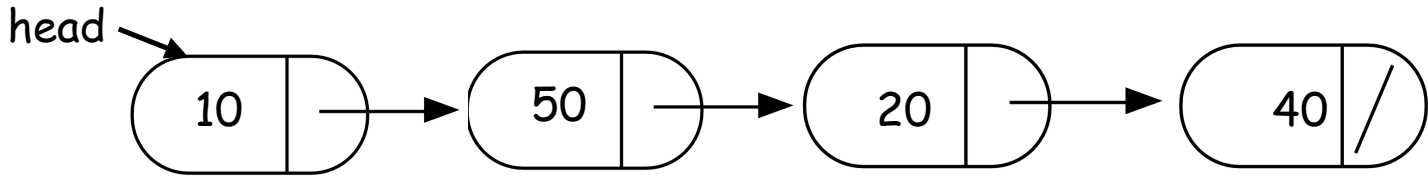


Describe a linked-list recursively



Which of the following methods of LinkedList CANNOT be implemented using recursion?

- A. Find the sum of all the values
- B. Print all the values
- C. Search for a value
- D. Delete all the nodes in a linked list
- E. All the above can be implemented using recursion



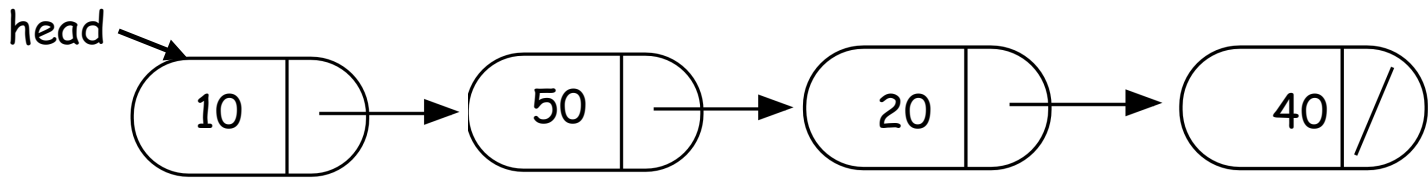
```
int IntList::sum(){  
    //Return the sum of all elements in a linked list  
}
```

# 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

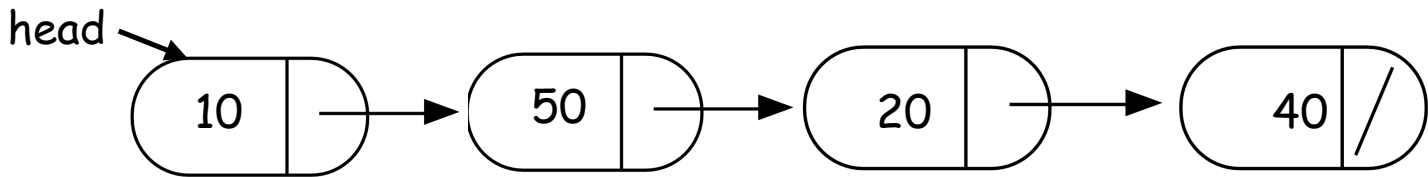
For example

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



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

```
}
```



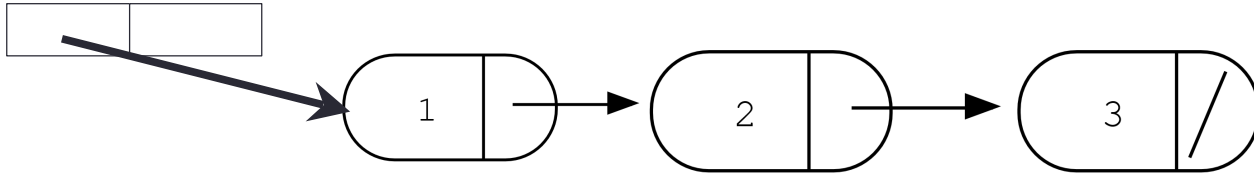
```
bool IntList::clear(Node* p){
```

```
}
```

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

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

head tail



# Next time

- Binary Search Trees