Course Introduction
Two variables
- \( x, y \)

Three operations
- \( x++ \)
- \( x-- \)
- \( (x=0)? L1:L2; \)

\begin{align*}
L1: & \quad x++; \\
    & \quad y--; \\
(y=0)? & L2:L1 \\
L2: & \quad \ldots
\end{align*}

Fact: This is “equivalent to” to every PL!

Good luck writing quicksort
... or Windows, Google, Spotify!
“A different language is a different vision of life”

- Federico Fellini
So Why Study PL?

Programming Language
Shapes
Programming Thought
So Why Study PL?

Programming Language

Affects How:

- Ideas are expressed
- Computation is expressed
Course Goals

“Free your mind”
- Morpheus
Learn New Languages / Constructs

New ways to:
- describe
- organize
- think about
computation
Write Programs That Are:

- Readable
- Correct
- Extendable
- Modifiable
- Reusable
To Learn ...
Learn How to Learn New PLs

No Java, C# 20 years ago

Learn the anatomy of a PL
• Fundamental building blocks
• Different guises in different PLs

Re-learn the PLs you already know
To Design New Languages
Learn How to Design New PLs

... “Who, me?”

Buried in every extensible system is a PL
- Emacs: Lisp
- Word, Powerpoint: Macros, VBScript
- Unreal: UnrealScript (Game Scripting)
- Facebook: FBML, FBJS
- SQL, Renderman, LaTeX, XML...
Choose Right Language

I WANT YOU
Learn How to Choose Right PL

“... But isn’t that decided by

• libraries,
• standards,
• and my boss?”

Yes!

My goal: educate tomorrow’s tech leaders & bosses, so you’ll make informed choices
Speaking of Right and Wrong...
Imperative Programming
x = x + 1
WTF?

x = x+1
Imperative = Mutation
Imperative: Mutation

Bad!
Don’t Take My Word For It

John Carmack
Creator of FPS: Doom, Quake,...

I am starting to remove op= operator overloads to discourage variable mutation.
Don’t Take My Word For It

Tim Sweeney (Epic, Creator of UNREAL)

“In a concurrent world, imperative is the wrong default”
Functional Programming
Functional Programming

No Assignment.
No Mutation.
No Loops.
OMG!
Who uses FP?!?
So, Who Uses FP?

MapReduce
So, Who Uses FP?

Linq, F#
So, Who Uses FP?

Facebook

Erlang
So, Who Uses FP?

Scala
So, Who Uses FP?

OCaml
So, Who Uses FP?

CSE 130!
Instructor

Ravi Chugh

TAs

Valentin Robert

Alexander Bakst

Tutor

Patrick Torbett

Office Hours TBD, Check Website ...
cseweb.ucsd.edu/classes/wi14/cse130-a/

Temporary link:
goto.ucsd.edu/~ravi/wi14/cse130-a/

- Nothing printed, everything on the web
- Feel free to bring a laptop to class
Discussion Sections

Wednesdays + Fridays
9:00am to 9:50 am
CSB 001

Quick Poll:

- Both “Review”
- One “Review”, One “Extra”
- Both “Extra”
Quick Poll:

How many people do **not** already have an i>clicker?
Clickers + Peer Instruction (ish)

• Make class interactive
  • Help YOU and ME understand what’s tricky

• Everyone must bring an i>clicker to class
  • by Thurs 1/16 (we’ll “practice” until then)
  • available at bookstore

• Seating in assigned groups
  • Will be posted to Piazza before next class
1. **Solo Vote**: Think for yourself, then vote

2. **Group Discussion**: Groups of ~3 students
   - Practice analyzing, talking about tricky notions
   - (Try to) reach consensus
   - If you have questions, raise your hand!

3. **Group Vote**: Everyone in group votes

4. **Class Discussion**:
   - What did you find easy/hard?
   - *Questions from here show up in exams*
In-Class Clicker Exercises

- Participation counts for 5% of your grade
- Respond to 75% of the questions throughout the quarter
- So, don’t fret if you miss a class or two
- Register your clicker! (check webpage)
Grading and Exams

- No Official “Before-Class” Homework
- In-Class Exercises: 5%
- Programming Assignments (6-8): 30%
- Midterm: 30%
- Final: 35%
Programming Assignments

Schedule up on webpage

Due on Fridays at 5:00 PM

You may use up to four late days total
• Each late day is a “single” or “whole unit”
• 5 mins late = 1 late day
• Plan ahead, no other extensions

PA #0 is online, not graded
Online lecture notes
Resources posted on webpage
Pay attention to lecture and section!
Do assignments yourself!
Plan for Next 10 Weeks

1. FP, OCaml, ~6 weeks
2. OO, Scala, ~4 weeks
Programming Assignments

Unfamiliar languages
+ Unfamiliar environments

Start Early!
No Compile, No Score
Forget Java, C, C++, ...  
... other 20th century PLs

Don’t complain ... that OCaml is hard  
... that OCaml is @!%@#  

Immerse yourself in new languages
Free your mind.
• Programming Assignments done **ALONE**

• We use *plagiarism detection* software
  • Have code from **all previous classes**
  • *MOSS* is fantastic, plagiarize **at your own risk**
while(mdis<sl){ /*showing values of the multiple between the limits*/
    printf("%d\n",mdis); /*10 spaces for each value, right align*/
    if((mdis%2)==1){ /*Counting odds and evens*/
        even++;
        rsun+=mdis;
        counter++;
        disMult+=mdis;
    }
    odd++;
    if(counter==6){ /*summing at end of each row*/
        printf("\n\n",rsun);
        sum+=rsun; /*running total sum by adding row sums*/
        rsun=0; /*reset row sum for next row*/
        counter1=1; /*reset counter for next row*/
    }
    if(counter==6){ /*summing at end of the last row*/
        printf("\n\n",rsun);
        counter1=1; /*reset counter for next row*/
    }
    printf("\n\nThere are %d odd and %d even numbers.\n",odd,even); /*stating the*/
    printf("The sum of all numbers is: %d\n\n",sum); /*showing the sum*/
    /*resetting the value for next time running through program*/
    even=0;
    odd=0;
    sum=0;
}
}
return 0;
• Programming Assignments done ALONE

• We use plagiarism detection software
  • Have code from all previous classes
  • MOSS is fantastic, plagiarize at your own risk

• Zero Tolerance
  • offenders punished ruthlessly

• Please see academic integrity statement

• Click Fraud is also not allowed!
To Ask Me Questions!
void sort(int arr[], int beg, int end){
    if (end > beg + 1){
        int piv = arr[beg];
        int l = beg + 1;
        int r = end;
        while (l != r-1){
            if(arr[l] <= piv)
                l++;
            else
                swap(&arr[l], &arr[r--]);
        }
        if(arr[l]<=piv && arr[r]<=piv)
            l=r+1;
        else if(arr[l]<=piv && arr[r]>piv)
            {l++; r--;}
        else if (arr[l]>piv && arr[r]<=piv)
            swap(&arr[l++], &arr[r--]);
        else
            r=l-1;
        swap(&arr[r--], &arr[beg]);
        sort(arr, beg, r);
        sort(arr, l, end);
    }
}
Say Hello to OCaml

Readability matters...

sort=:({~ ?@#) {~ ?@#)^: (1:<#)

Quicksort in J
let rec sort xs =
  match xs with
  [] -> []
  | h::t ->
  let (l,r) = List.partition ((<=) h) t in
  (sort l) @ h::(sort r)
Plan for Next ~5 Weeks

1) Fast Forward
   • Rapid introduction to what’s in OCaml

2) Rewind

3) Slow Motion
   • Go over the pieces individually
CSE 130 [Winter 2014] Programming Languages

Introduction to OCaml

Ravi Chugh

UCSD CSE
Computer Science and Engineering
“Meta Language”
Designed by Robin Milner
To manipulate theorems & proofs

Several dialects:
• Standard ML (SML)
  • Original syntax
• Objective Caml: (OCaml)
  • “The PL for the discerning hacker”
  • State-of-the-art, extensive library, tool, user support
• F# (ML + .NET) released in Visual Studio
ML’s Holy Trinity

- Everything is an expression
- Everything evaluates to a value
- Everything has a type
Interacting with ML

“Read-Eval-Print” Loop (REPL)

Repeat:
1. System reads expression $e$
2. System evaluates $e$ to get value $v$
3. System prints value $v$ and type $t$

What are these expressions, values and types?
Complex expressions using “operators”: (why the quotes?)

- +, -, *
- div, mod
Complex expressions using “operators”: (why the quotes ?)

- Concatenation ^
Complex expressions using “operators”:

- “Relations”: =, <, <=, >=
- &&, ||, not

```
true  
false  
1 < 2  
“aa” = “pq”

(true) && (false)  
(true) && (true)  
(false) && (true)

bool
```
Untypable expression is rejected

- No casting, No coercing
- Fancy algorithm to catch errors
- ML’s **single most powerful** feature (why?)
1. Enter an expression $e$
2. ML infers a type $t$ or emits an error
3. ML evaluates expression $e$ down to a value $v$
4. Value $v$ is guaranteed to have type $t$
So Far, A Fancy Calculator...

... What do we need next?
Complex Type: Tuples (Products)

(2+2 , 7>8);  \rightarrow  (4,\text{false})

\text{int} \times \text{bool}
• Pairs, Triples, Quadruples, ...
• Nesting:
  - Everything is an expression
  - Nest tuples in tuples
• **Unbounded size**
• **Can have lists of anything (e.g. lists of lists)**
• **But ...**
Complex Type: Lists

All elements **must have same type**

```
[1; "pq"];
```
Complex Type: Lists

List operator “Cons” ::

1::[]; // int list
1::[2;3]; // int list
"a"::["b";"c"]; // string list

Can only “cons” element to a list of same type

1::["b"; "cd"];
Lists: “Cons”truct

Nil operator

Cons operator
List operator “Append” @

\[
[1; 2] @ [3; 4; 5];
\]

\[
[\text{"a"}] @ [\text{"b"}];
\]

\[
[] @ [1];
\]

\[
1 @ [2; 3];
\]

\[
[1] @ [\text{"a"}; \text{"b"}];
\]

Complex Type: Lists

Can only append two lists...

... of the same type
List operator “Head”  \texttt{List.hd}

\begin{itemize}
  \item \texttt{hd [1;2];} \quad \rightarrow \quad 1 \quad \textbf{int}
  \item \texttt{hd ("a"@["b"]);} \quad \rightarrow \quad "a" \quad \textbf{string}
\end{itemize}

\textbf{ML types can’t catch some errors though...}

\begin{itemize}
  \item \texttt{hd []} \quad \textbf{Exception: Failure "hd".}
\end{itemize}

\textit{(ML does infer a type...)}
Complex Type: Lists

List operator “Tail” \texttt{List.tl}

\begin{align*}
\text{tl} \ [1;2;3]; & \quad \Rightarrow \ [2;3] \\
\text{tl} \ (\text{"a"}@\text{"b"}); & \quad \Rightarrow \ \text{"b"}
\end{align*}

The tail of empty list is a run-time error...

\texttt{tl} \ [\] \quad \text{Exception: Failure “hd”}.

Expressiveness of type systems is an active area of research!
Lists: Deconstruct (or Destruct)

Head

\[ e : T \text{ list} \]
\[ \text{hd } e : T \]
\[ e \Rightarrow v_1 \]
\[ \text{hd } e \Rightarrow v_1 \]

Tail

\[ e : T \text{ list} \]
\[ \text{tl } e : T \text{ list} \]
\[ e \Rightarrow v_1 :: v_2 \]
\[ \text{tl } e \Rightarrow v_2 \]

\((\text{hd } [[], [1; 2; 3]]) = (\text{hd } [[], ["a"]])\)

int list

\[ e_1 : T \]
\[ e_2 : T \]
\[ e_1 = e_2 : \text{bool} \]

string list
Recap: Tuples vs. Lists

What’s the difference?

• Tuples:
  - Different types, but fixed number:
    - pair = 2 elts
      (3, “abcd”) (int * string)
    - triple = 3 elts
      (3, “abcd”, (3.5, 4.2)) (int * string * (float * float))

• Lists:
  - Same type, unbounded number:
    [3;4;5;6;7] int list

• Syntax:
  - Tuples = comma  Lists = semicolon