Computer Science 414: Compilers

Home
Syllabus
Lecture Notes
Assignments

David Galles
Computer Science
Univerisity of San Francisco

Syllabus for CS414 Compilers for Spring 2017

Time: MWF 11:45-12:59

Location: LS G12

Professor: David Galles
Office: HR 542
Office Hours**:**T /TR 10-12 or by appointment
Though these are my stated office hours, I am in my offices most of the day.
If my door is open (and it usually is), I am happy to talk with students.
Phone: 422-5951

Email:galles@usfca.edu

Textbook: *Modern Compiler Design* D. Galles

Prerequisites:

Computer Science 245, Data Structures and Algorithms
Computer Science 220 or Computer Science 221
Computer Science 315 (***recommended*** prerequsite or corequisite)

Test Dates:

Test	Weight	Test Date
Projects	50%	Various
Midterm #1	15%	03/13/2017
Midterm #2	15%	04/26/2017
Final:	20%	5/13/2017 (Saturday) 12:30pm

Finals and Midterms:
Both midterms and the final will be closed notes.

Projects:
You must submit an electronic version of your source code via subversion.
NOTE -- To pass the class, your final project needs to meet the minumum requirements in the Grading section

Academic Dishonesty:
Any instances of academic dishonesty (cheating on an exam, code plagarization, etc) will result in an automatic zero on that assignment or test, and referral to the deans office.

Late Policy:
Late projects will be accepted on the next class meeting after the due date for up to 75% credit. Projects will not be accepted later than one class meeting after the deadline.

Grading:
Grades will be assigned on a straight scale, with *Approximately*

90-100% A
80-89% B
65-79% C
55-64% D
0-54% F

In addition to the above percentages, your final compiler must correctly compile the following code to obtain a C in the class (thus if your program cannot compile this file, you will get less than a C, and this class **will not count towards the CS major**):

```
void main()
{
    int inputVar;
    inputVar = read();
    inputVar = inputVar + 1;
    print(inptVar);
}
```

In addition to the above percentages, your final compiler must correctly compile the following code (which solves the n-queens problem) to obtain an A in the class:

```
void PrintBoard(int board[], int size);
int Abs(int x);
boolean Legal(int numcols, int board[]);
void Solve(int numcols, int board[], int size);

void main() {
   int size;
   int board[];

   size = Read();
   board = new int[size];

   Solve(1, board,size);
}

void PrintBoard(int board[], int size) {

   int i;
   int j;

   for (i=0; i<size; i++) {
      for (j=0; j<board[i]; j++)
         Print(0);
      Print(1);
      for(j=board[i]+1; j<size; j++)
         Print(0);
      Println();
   }
   Println();
}


int Abs(int x) {
   if (x >= 0)
      return x;
   else
      return  0-x;
}
```

```
boolean Legal(int numcols, int board[]) {
   int i;
   int j;
   boolean legal;

   legal = true;


   for (i=0; i < numcols; i++) {
      for (j=i+1; j < numcols; j++) {
         if ((board[i] == board[j]) ||
             (Abs(i-j) == Abs(board[i]-board[j])))  {
            legal = false;
         }
      }
   }
   return legal;
}

void Solve(int numcols, int board[], int size) {
   int i;


   if (numcols > size)  {
      PrintBoard(board,size);
      Println();
   }
   else
      for (i=0; i < size; i++) {
         board[numcols-1] = i;
         if (Legal(numcols, board)) {
            Solve(numcols+1, board, size);
         }
      }
}
```

Finally, you can receive a 1/2 grade bonus (B+ to A-, A- to A, and so on) by extending your compilier to allow simple object oriented programming.

```
class Point
{
    int x;
    int y;
    Point(int xval, int yval)
    {
        x = xval;
        y = yval;
    }
    void setX(int xval)
    {
        x = xval;
    }
    void setY(int yval)
    {
        y = yval;
    }
    int getX() {
        return x;
    }
    int getY()
```

```
    {
        return y;
    }
    void Print()
    {
        print(x);
        print(y);
        println();
    }
}
void main()
{
    int x = read();
    int y = read();
    Point p1 = new Point(x,y);
    Point p2 = new Point(y,x);
    p1.Print();
    p2.Print();
    p1.setX(y);
    p1.setY(x);
    p1.Print();
    p2.Print();
}
```

Attendance:

Students are expected to attend class. Topics that are discussed in class but are not in the text are fair game for the midterms and final. While I will make an effort to make as much of the class material as possible available online, there will likely be some information that you will only be able receive by attending class.

Learning Outcomes:

Students who complete this course will be able to:

1. Use regular expressions
2. Design an EBNF for a language, given an informal description of the language
3. Build a simple top-down (LL(1)) parser by hand for a simple language
4. Build a simple bottom-up (LR(1), SLR(1)) parser by hand for a simple language
5. Understand the complete compilaion process, including lexical analysis, parsing, semantic analysis, and code generation
6. Build a complete compiler using parser generator tools (javacc)