SW483, Summer 2017
CS131 Homework Assignment #7, “The Date Printer”, in Lisp!

Compiling and Running an Incomplete DatePrinter Program

First, download, compile, and run a simple, but incomplete and incorrect, date printer program in Lisp. This program takes as input two integers from the console and prints out the corresponding date to the console. You can compile, run, and test the incomplete calculator program using the same steps as for previous assignments. There is a zip archive containing the code (date_printer.lisp).

To compile and run the date printer program:

1. Download DatePrinter.zip. It contains a Lisp source code file, date_printer.lisp, a bash shell script, assignment_7_grader.sh, and a file with test cases: test_cases.csv.

2. Use the bash shell and change directories to the folder created by unpacking the .zip file.

3. Then simply execute “clisp date_printer.lisp” and follow the instructions in the console to run the program.

Completing the DatePrinter Program

For this assignment, you will implement a program that takes two integers and prints out the corresponding month name and day to the console. For example

```
Enter two integers.  Press return after each integer.
10
4
October 4th
```

Your program should print the name of the month, correctly spelled and capitalized. Then it prints the day of the month, and also prints two letters corresponding to the ordinal pronunciation of the day, in this case “th” for “fourth.” There is no space between “4” and “th”, but there is a space between the name of the month, “October” and the “4”. The entire date should be printed on a single line of text after the user enters the second number, and is not split up over multiple lines.

The ordinal numbers are written as follows: 1st, 2nd, 3rd, 4th, 5th, 6th, 7th, 8th, 9th, 10th, 11th, 12th, 13th, 14th, 15th, 16th, 17th, 18th, 19th, 20th, 21st, 22nd, 23rd, 24th, 25th, 26th, 27th, 28th, 29th, 30th, 31st. (Ignore the fact that in the previous sentence, the suffixes are superscripted, small, and raised. In your program they are printed to the console just like any other text.)

Because the user enters integers to the console, there is always a possibility that the user enters two integers that specify a date that doesn't exist. In this case, rather than print the date, your program prints the text "Invalid Date". For example:

```
Enter two integers.  Press return after each integer.
15
-200
Invalid Date
```

A date is considered invalid if any of the following conditions is met:

1. The month specified is 0 or a negative integer

2. The month specified is 13 or greater.

3. The day specified is 0 or a negative integer.

4. The day specified doesn't exist for the corresponding month. For example April only has 30 days. 31 is an invalid day for April, and so is any day greater than 31. Assume the normal number of days for each month in the standard (Gregorian/Western) calendar. Assume that it is a leap year, and that February has 29 days instead of the usual 28.

Note that when the program prints "Invalid Date", nothing else appears on the console before the program terminates. No incomplete months or dates should appear in this case.

We suggest doing this assignment in two parts. First, compose some code that determines whether the integers entered specify a valid or invalid date. If the date is valid, this code prints the month name and the day. If the date is invalid, it prints "Invalid Date". Second, compose some code that prints the ordinal suffix, but coordinate it with the first part of the code so that it only prints the suffix when the date is valid.

Checking for Validity and Printing the Month

Some functionality in your code will need to check for validity of the date. As part of checking the validity (or separate from it, if you like), your code will also need to print the month name and the day. There are several ways to accomplish this. We suggest one way below, as well as other options you may want to pursue.

One way, which is suggested by the code given to you, is to have a cond expression that checks the month and the day together for validity. This is necessary because not all months have the same number of days.

If the month and day are valid, then the code prints out both the month and the day using "format". This allows a separate part of the code in the program to determine the correct ordinal suffix ("st", "nd", "rd", or "th") so that it can be printed right after the day. If the month and day are not valid, then the code prints "Invalid Date".

You can complete the code that is given to you by adding more s-expressions to the "cond", one corresponding to each month. One way to do this is to copy and paste one of the s-expressions given to you into the cond as many times as needed. You will have to change the sub-expressions of each copy to correspond to a particular month and the number of days in that month.

Printing the Ordinal Suffixes

You will also need to add a separate part to your code that handles the ordinal suffixes. The suffix should be printed after the day, with no intervening space.

The only code we've given you to start with prints "st" in the right situations, but always prints the suffix "th" otherwise, which is obviously incorrect. There are several different ways to print the ordinal suffixes. One way is to put a cond that prints "st" if day is 1, "nd" if day is 2, "rd" if day is 3, and so on, up to "st" when the day is 31. You will need 31 s-expressions in the cond to do it this way. Another option is to use a "case" instead of a "cond." This method can work, but it is a very repetitive "brute force" method. Also, remember that you only want to print the suffix when the date is valid, so you'll also have to add code for that. We encourage you to see the next section for suggestions about other implementation options that may be more terse or elegant.

Exploring Other Implementation Options

The ways we have suggested so far aren't the only possible ways of doing this assignment. You may want to consider or experiment with these other options for achieving the same functionality.

Here are some other options that you may want to explore for printing the ordinal suffix:

1. You'll notice that the code you've been given for printing the ordinal suffix is has a cond and an s-expression where the "test expression" is "(member day '(1 21 31))" and the "action" expression prints "st". The "member" expression evaluates the day and checks to see if its value exists in the list,

"'(1 21 31)". Therefore if the day is 1, 21 or 31, it prints "st". You may add more similar s-expressions to the cond to handle the other suffixes.

2. Notice also that the second s-expression has the test expression "T". This means that it acts as a "default case" that always triggers if none of the other s-expressions in the cond work. It prints "th", the most common suffix. As long as you put s-expressions corresponding to "nd", and "rd" before it, you won't need to specify all of the numbers that have "th" as a suffix.

3. You will need to make it so that the ordinal suffix is only printed when the date is valid. If we don't do this, we'll see the output "Invalid Dateth" or just a "th" on the last line. One way to do this is to make it so that the earlier cond that checks for the validity of the date assigns a symbol the value T if the date is valid, and NIL if it is not. This symbol could be "dateIsValid" or something similar. Then the cond expression that prints the suffix can be wrapped inside of an "if" expression with your variable dateIsValid as the "test clause".

4. Another way to do this is by moving the code that figures out the ordinal suffix to be ahead of the code that prints, the date. Then, instead of the ordinal suffix code printing the suffix, it assigns a symbol to be a string with the correct suffix. Then the code that checks to see if the date is valid simply prints everything, the month, day, and the ordinal suffix. This way you don't need an extra "dateIsValid" symbol or extra checks for the valid date.

For writing your code that detects whether the date is valid and prints the month and day, you might want to consider these options:

1. Use "member" expressions for the number of days in a month. Because many months have the same number of days, you may be able to group the 30-day months and the 31-day months together and have a single shared chunk of code that checks the day to see that its valid.

2. You can check for date validity separately from printing the month and the day. This may work well with option (1) above.

3. Use a "case" expression instead of a "cond" to print the month.

4. Define a function that takes an integer as an argument and prints the month name or evaluates to the month name. Use that function.

Part 3, Running the Autograder Script

Run the autograder script by executing "./assignment_7_grader.sh" from the bash shell in the zip directory. Your Lisp file must be named "date_printer.lisp" with a lower-case "d" and all other letters lower-case.

The script will attempt to compile your date_printer.lisp and will test your code by executing it using the clisp interpreter on a number of inputs. When it completes, the autograder script will print your final score (out of 100) to the console. Ignore any compiler warnings.