Assignment Prefix: Lab10
Points: 100
Due Date: Friday, November 11, 2016 @ 11:59pm
This is an individual assignment.

## Task:

Write a Java project that:

- Implements the cyclic-shift hash code computation described in the text.
- Calculates and displays the cyclic-shift hash codes for the strings:
  - POTS
  - STOP
  - TOPS
  - This set of strings should be hard-coded into your program.
- Implements a verbose version of the cyclic-shift hash code computation that shows the 32-bit integer bit pattern at each stage of the cyclic-shift hash code computation ( see example below ).
- Displays the verbose cyclic-shift hash code computation for the above strings.

## Turning in your assignment:

- **Make sure that all of your code is properly documented.**
- Turn in your assignment using the standard method.
- Create a Word document using the standard naming convention:
  - Lab10-LnameF.docx
- Copy and paste each of your Java files into the document.
- Paste the screenshots showing the complete output of a complete run of your program after the Java code in your document.
- Export your NetBeans project to a zip archive.
- Turn in the Word document and zipped project as to separate files in a single Blackboard submission.

## Example Output ( only shows verbose for one string ):

```
Summary hash code information:


      POTS : 00000000001010010100011011010011

      STOP : 00000000001010101101101000110000

      TOPS : 00000000001010110100011001010011



Detailed hash code information:


Creating hash code for POTS:


Entering hashCode, pass  0     00000000000000000000000000000000


hashCode <<5                   00000000000000000000000000000000

hashCode >>> 27                00000000000000000000000000000000

hashCode <<5 | hashCode>>>27   00000000000000000000000000000000


Adding Character P             00000000000000000000000001010000


Exiting hashCode               00000000000000000000000001010000


Entering hashCode, pass  1     00000000000000000000000001010000


hashCode <<5                   00000000000000000000101000000000

hashCode >>> 27                00000000000000000000000000000000

hashCode <<5 | hashCode>>>27   00000000000000000000101000000000
```

```
Adding Character O             00000000000000000000000001001111


Exiting hashCode               00000000000000000000101001001111


Entering hashCode, pass  2     00000000000000000000101001001111


hashCode <<5                   00000000000000010100100111100000

hashCode >>> 27                00000000000000000000000000000000

hashCode <<5 | hashCode>>>27   00000000000000010100100111100000


Adding Character T             00000000000000000000000001010100


Exiting hashCode               00000000000000010100101000110100


Entering hashCode, pass  3     00000000000000010100101000110100


hashCode <<5                   00000000001010010100011010000000

hashCode >>> 27                00000000000000000000000000000000

hashCode <<5 | hashCode>>>27   00000000001010010100011010000000


Adding Character S             00000000000000000000000001010011


Exiting hashCode               00000000001010010100011011010011


hash code for POTS is          00000000001010010100011011010011
```