Lab 2: If I were a...

Due date: Thursday, October 9, 11:59pm.

Lab Assignment

Assignment Preparation

Lab type. This is an **pair programming lab**. The pairs will be designated by the instructor at the beginning of the lab. See the pair programming handout from the first day of classes for more information on what is expected of you.

Collaboration. Students work in pairs, and it is considered cheating, if members of the team (pair) do not work together. Communication between pairs during lab time is allowed, but no direct sharing of code is allowed.

Purpose. The lab allows you to practice the use of conditional statements and see. It also facilitates learning the use of boolean expressions as conditions in `if` and `switch` statements.

Programming Style. All submitted C programs must adhere to the programming style described in detail at

http://users.csc.calpoly.edu/∼cstaley/General/CStyle.htm

When graded, the programs will be checked for style. Any stylistic violations are subject to a 10% penalty. Significant stylistic violations, epsecially those that make grading harder, may yield stricter penalties. Also note the the Lab 2 requirement for the content of the header comment in each file you submit applies **to each assignment** (lab, programming assignment, homework) in this course.

Testing and Submissions. Any submission that does not compile using the

```
gcc -ansi -Wall -Werror -lm
```

compiler settings will receive an automatic score of 0.

For each program you have to write, you will be provided with instructor's executable and with a battery of tests. The programs you submit must pass all tests made available to you. You can check whether or not a program produces correct output by running the instructor's executable on the test case, then running yours, and comparing the outputs.

Program Outputs must co-incide. Any deviation in the output is subject to penalties (e.g., use of different words in the output). The exception is made in case of floating point computations leading to differences in the last few decimal digits.

Please, make sure you test all your programs prior to submission! Feel free to test your programs on test cases you have invented on your own[1].

The Task

Note: Please consult the instructor if any of the tasks are unclear.

In this lab you, working in pairs, will perform the following tasks:

1. Fix all errors in your Lab 2 submission (this includes **C program style issues** as well.
2. Change your Lab 2 programs to protect them from incorrect input values.
3. Write a few programs in the If-Statement Zones world.
4. Create the next version of your electoral vote computation program, which will, now, (a) take as input popular vote tally in each state, (b) determine the winner of each state and (c) determine the winner of the electoral college vote.

Task 1: Perfecting Lab 1 programs.

This task introduces you (in a somewhat simplistic way) to the concepts of maintenance (making changes and fixing problems in a program after its original deployment) and code reuse (use of code written for a previous task in your solution of the new set of problems).

Your original Lab 1 programs could take as input numbers that would not make sense given the problem specification. For example, in the currency exchange program, commission was supposed to be between 0 and 100, but nothing prevented the user from entering -5 or 160. *The tests for Lab 1 programs ignored incorrect inputs*, but in reality, you have to **protect** your inputs and make sure that they are meaningful.

In this task, you will revisit the attendance percentage and the currency converter programs, and will rewrite them to ensure that you check for correctness of all inputs.

Genearal Requirement. This is a pair programming assignment, so only one submission is required. For each of the three programs you only need to

[1] In this lab, we provide you with a large collection of test cases. In some future labs, test case development will be a major part of the lab assignment, so it never hurts to start early

extend the code of one of the two versions that each pair has access to. The second version need not be modified.

Fix the Bugs!

Before starting to change your programs, you shall inspect all Lab 1 submissions and fix all problems found in them by the grading. This task is to be performed **in pairs** on the programs of **both team members**! Fix all the following:

1. Any noted by the instructor/grader faults in your program (a fault is a logical error in your program - e.g., incorrect order of statements, or incorrect computation);
2. Any noted by the instructor/grader errors in the form of the output (incorrect wording, spacing, extra or missing line breaks);
3. Any noted by the instructor/grader errors in program in style (indentation, variable naming, line length, comments, use of constants).

Remember to do it for the programs of **both** team members.

Program 1: School attendance

New requirements.

A1. The input to the program, the number of classes skipped **must be** an integer value between 0 and the total number of lectures during the quarter.

A2. After the input is read, your program must check if the number obtained from the standard input is in the range prescribed by **A1.**. If it is, the program shall proceed as before. If it is not, the program shall print

```
Incorrect Input: <Skipped>
```

where `<Skipped>` is the value obtained from the standard input. After that the program shall stop execution.

A3. Rename your old (Lab 1) program to `attendance-old.c` Name the new version of your program `attendance.c`.

Program 2: Currency converter

New Requirements.

C1. The program has three inputs: the exchange rate, the commission, and the amount of money to be changed. The following are the acceptable value ranges for each of the input values:

- exchange rate: *positive* (not 0!).
- commission: *non-negative* (0 is OK!), less than 100.
- dollars to be exchanged: *non-negative.*

C2. After each of the three inputs is read from the standard input, your program shall check if the value read is in the range prescribed by **C1.**. If it is, the program shall proceed as before. If it is not, the program shall output the following:

- If the exchange rate was incorrect:

  ```
  Incorrect exchange rate: <Rate>
  ```

  where `<Rate>` is the incorrect exchange rate number read from the standard input.

- If the commission was incorrect:

  ```
  Incorrect commission: <commission>
  ```

  where `<commission>` is the incorrect commission read from the standard input.

- If dollars to be exchanged was incorrect:

  ```
  Incorrect dollars: <Dollars>
  ```

  where `<Dollars>` is the incorrect amount of dollars to be exchanged read from the standard input.

After that, the program shall stop its work.

C3. Rename the old (Lab 1) version of your program `converter-old.c` Name then new version of your program `converter.c`

Task 2: New Electoral College Program

This part of Lab 3 asks you to produce a new version of the electoral college program. While time is still not right to simplify the program and make it shorter and less repetetive, you **can now** make the program a bit more **informative**.

The new functionality of your program is described below.

EC1. Your program will now run in one of two "modes", verbose or silent. Informally, in verbose mode, your program will output election results for each state, as well as the overall election results. In silent mode, your program will behave similarly to its behavior in Lab 2: only the overall election results will be reported.

EC2. The input to your program now shall be a **sequence of 52 zeroes or ones**. The first number in the sequence is the **mode indicator**, while the remaining 51 numbers remain as before the results of elections in each of the states and DC provided in the alphabetical order of the state name.

EC3. If the **mode indicator** (the first input of the program) is equal to **0** then the program shall run in the silent mode.

If the **mode indicator** is equal to **1**, the program shall run in the verbose mode.

EC4. Your program shall start by initializing `votesMcCain` and `votesObama` as prescribed by requirement **ECF3** (see Lab 2). After that, your program shall read the **mode indicator**[2].

EC5. After that, your program shall proceed as before (see **ECF4.** from Lab 2), with the following changes to its behavior after each `stateDecision` is read:

EC5-1. Your program shall check if the **mode indicator** points to verbose or silent mode of running.

EC5-2. If the **mode indicator** points to silent mode, your program proceeds as before (see **ECF4.** from Lab 2).

EC5-3. If the **mode indicator** is set to verbose mode, your program shall perform the following actions:

- Check if the current state had been won by McCain or Obama.
- If Obama has carried the state, print

  ```
  <State> goes to Obama
  ```

  where `<State>` is the name of the current state (`Alabama, Alaska, Arizona`, etc...).
- if McCain has carried the state, print

  ```
  <State> goes to McCain
  ```

After outputting the text as above, your program shall proceed reading the decision of the next state as prescribed by **ECF4** of Lab 2.

EC6. After the information about the decisions of all states is read from standard input, your program shall compute and output the electoral college votes for Obama and McCain as prescribed by requirement **ECF5, ECF6** of Lab 2. After that, your program shall determine the winner of the electoral college.

Remember, that the winner of the elcoral college is the candidate who won the majority of the electoral college vote. At present, the Electoral College admits three possible results:

- Barack Obama gains 270 or more elecotral college votes and wins the Electoral College outright (and gains more Electoral College votes than John McCain). In this case, your program shall output

  ```
  The next President of the United States is Barack Obama.
  ```

- John McCain gains 270 or more electoral college votes and wins the Electoral College outright (and gains more Electoral College votes than Barack Obama). In this case, you program shall output.

  ```
  The next President of the United States is Barack Obama.
  ```

[2]Please note, that **mode indicator** is a new piece of data your program will work with, therefore, a new variable to store it shall be declared. In the future, it is left up to you to understand when new variables need to be declared in similar situations.

- Both John McCain and Barack Obama gain 269 Electoral College votes[3]. According to the Constitution, Electoral College ties are broken in the House of Representatives. As the Democrats currently hold the majority there [4], it is likely, although not 100% that Barack Obama will become the next President. If a tie case is detected by your program, it shall output the following text:

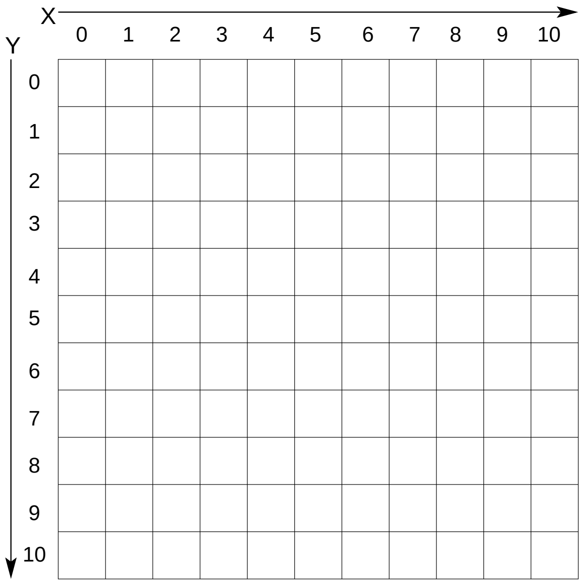
  ```
  It's an Electoral College tie! The next President is probably Barack Obama.
  ```

EC7. Name your program `elections.c`.

Task 3: If-Statement Zones

If-Statement Zones are regions of a 11×11 two-dimensional grid. The final task of the lab asks you to **recognize** specific if-statement zones from the examples provided to you.

The overall grid, on which the if-statement zones will be marked is shown below:



Each zone on the grid has a color. The color of the zone comes from the following list (note, all color names are lowercase):

```
red
green
blue
black
yellow
purple
orange
```

[3] A situation which is quite possible, should Obama win all "Kerry states" except for New Hamphire, as well as Iowa, New Mexico and Colorado.

[4] The exact procedure of voting is somewhat more complicated, but the Democratic party also holds the majority of state delegations as well.

Your task is, given a grid with a number of zones write a program which determines color of each point on the grid.

General Requirements

Z1. If a region has no color, it is assumed blank, your program shall provide no output. There is no "white" color.

Z2. For each program you are given the exact specification of **how** the decisions about the colors need to be made. In particular, we specify the number of `if` statements, the number of `&&` and the number of `||` logical operators you can use in the program. (Note, in this lab, you are only allowed the use of `if` statements, no `&&` or `||` operators). You may use as many comparison (==, !=) and logical negation (!) operators as well as as many parenthesis as you like.

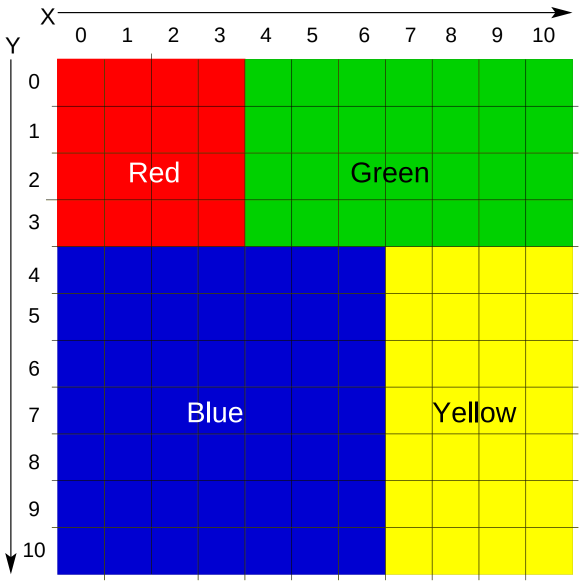
Z3. Never use an empty `if` or `else` branch in your code ("empty" means a branch with no statements in it).

Z4. All programs shall work in exactly the same manner. The x and y coordinates on the grid are `integers`. The program shall prompt for the x coordinate first (the prompt text is `Enter X:`), read the x value, then prompt for the y coordinate (the prompt texts is `Enter Y:`), read the y value. After that, each program needs to perform a series of checks to establish the color of the given point. Once the color is determined, it is to be printed out, after which the program should stop its operation.

Program 1: zones01.c

Restrictions:

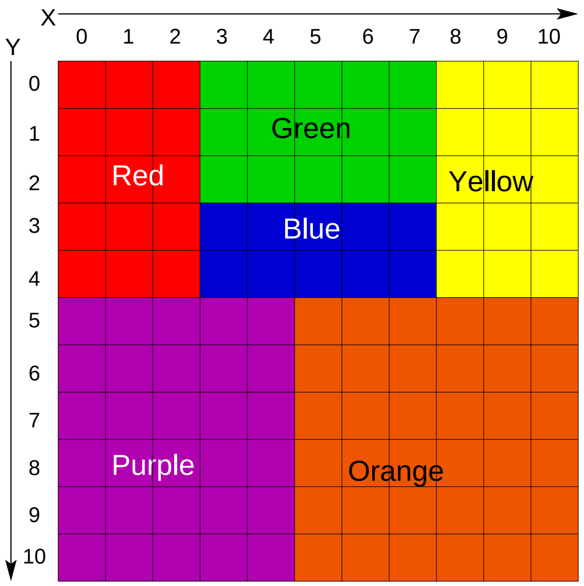
Number of `if`s:	3
Number of `&&` operators:	0
Number of `||` operators:	0



Program 2: zones02.c

Restrictions:

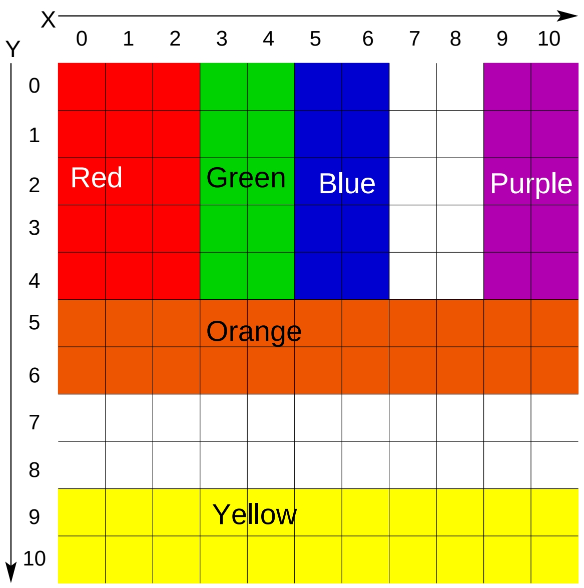
Number of `ifs`:	5
Number of `&&` operators:	0
Number of `||` operators:	0



Program 3: zones03.c

Restrictions:

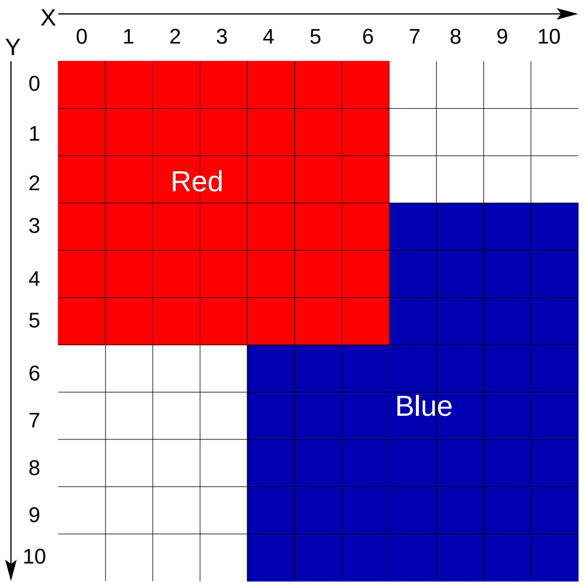
Number of `ifs`:	7
Levels of indentation:	4
Number of `&&` operators:	0
Number of `||` operators:	0



Program 4: zones04.c

Restrictions:

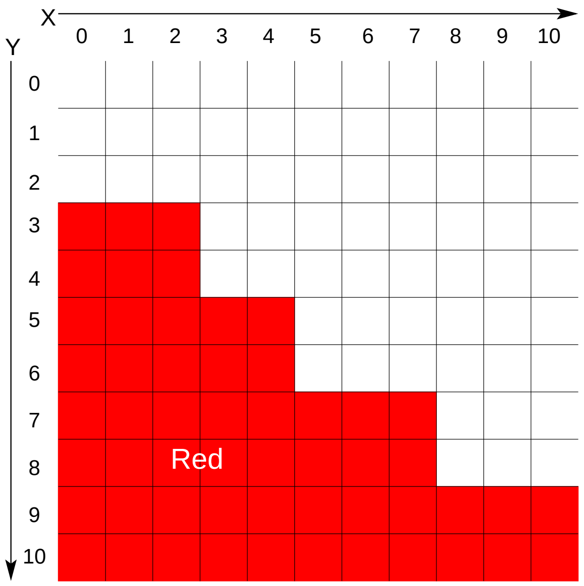
Number of `ifs`:	4
Number of `&&` operators:	0
Number of `||` operators:	0



Program 5: zones05.c

Restrictions:

Number of `ifs`:	7
Number of `&&` operators:	0
Number of `||` operators:	0



Submission.

Files to submit. You shall submit **nine** files:

> `team.txt`, `attendance.c`, `converter.c`, `elections.c`, `zones01.c`,`zones02.c`, `zones03.c`, `zones04.c`, `zones05.c`.

`team.txt` file shall contain the names of the two team members in each pair, and the Cal Poly IDs of each. E.g, if I were on the team with Dr. John Bellardo, my `team.txt` file would be

```
John Bellardo, bellardo
Alex Dekhtyar, dekhtyar
```

No other files can be submitted. In fact, if you submit *other* file, or submit one of the three files about with an *incorrect filename*, you will receive an email informing you about a submission error, and asking you to resubmit.

Files can be submitted one-by-one, or all-at-once.

Submission procedure. As with Lab 1, you will be using `handin` program to submit your work. The procedure is as follows:

- `ssh` to `vogon` (`vogon.csc.calpoly.edu`).
- Students from **Section 009** shall execute the following submission command:

  ```
  > handin dekhtyar lab03-09  <your files go here>
  ```

- Students from **Section 011** shall execute the command:

  ```
  > handin dekhtyar lab03-11 <your files go here>
  ```

`handin` is set to stop accepting submissions 24 hours after the due time.

Grading

The lab grade is formed as follows:

`attendance.c`	5%
`converter.c`	5%
`elections.c`	15%
`zones01.c`—`zones05.c`	15% each

Any submitted program that does not compile earns 0 points.

Any submitted program that compiles but fails at least one **public** (i.e., made available to you) test earns no more than 30% of its full score (and can possibly earn less).

Any submitted program that compiles and succeeds on **all** publically available tests earns at least 50% of its full score.

All programs will be checked for style conformance. Any style violation will be noted. The program will receieve a 10% penalty.

Appendix A. Testing

Instructor's executables. Instructor's executables for all programs are provided to you on the course web page. When copying them, please make sure to run `chmod u+x` on them.

Public test suites. Public test suite for `attendance.c`, `converter.c` consists of the public tests from Lab 2, plus extra tests, made available to you by the instructor. The tests can be downloaded as a gzipped tar file from the course web page.

Public test suite for `elections.c` is new. Each Lab 2 test file had been checked for correctness. In addition, one more number has been entered at the beginning of each test case to indicate the mode (silent/verbose). The tests can be downloaded as a gzipped tar file from the course web page.

All If-statement zone programs are tested using the same test suite, made available to you on the course web page.

Test Runs. Below are the test runs of all the programs you have to write. Note, the outputs were copied from *real runs* of the instructor's executables.

`attendance.c`

```
>attendance
How many classes skipped? 4

The number of lectures in your course is 21.000000
You skipped 19.047619 % of your classes. Beware!
>attendance
How many classes skipped? -2

The number of lectures in your course is 21.000000
You skipped -9.523809 % of your classes. Beware!
> attendance
How many classes skipped? 23

The number of lectures in your course is 21.000000
You skipped 109.523811 % of your classes. Beware!
```

`converter.c`

```
>converter
Currency Converter Kiosk: USD to RR

Enter exchange rate: 0
Incorrect exchange rate: 0.000000
> converter
Currency Converter Kiosk: USD to RR

Enter exchange rate: -4
Incorrect exchange rate: -4.000000
> converter
Currency Converter Kiosk: USD to RR

Enter exchange rate: 1
Enter commission%: -1
Incorrect commission: -1.000000
> converter
Currency Converter Kiosk: USD to RR

Enter exchange rate: 3
```

```
Enter commission%: 3
How much money do you want to change? 0
Incorrect dollars: 0
```

elections.c

```
> cat election-test01
0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
> elections < election-test01
Electoral College vote total for John McCain (R) is 538
Electoral College vote total for Barack Obama (D) is 0
The next President of the United States is John McCain
> cat election-test05
1
1 0 1 0 1 0 1 0 1 0
1 0 1 0 1 0 1 0 1 0
1 0 1 0 1 0 1 0 1 0
1 0 1 0 1 0 1 0 1 0
1 0 1 0 1 0 1 0 1 0 1
> elections < election-test05
Alabama goes to Obama
Alaska goes to McCain
Arizona goes to Obama
Arkansas goes to McCain
California goes to Obama
Colorodo goes to McCain
Connecticut goes to Obama
Delaware goes to McCain
District of Columbia goes to Obama
Florida goes to McCain
Georgia goes to Obama
Hawaii goes to McCain
Idaho goes to Obama
Illinois goes to McCain
Indiana goes to Obama
Iowa goes to McCain
Kansas goes to Obama
Kentucky goes to McCain
Louisiana goes to Obama
Maine goes to McCain
Maryland goes to Obama
Massachussets goes to McCain
Michigan goes to Obama
Minnesota goes to McCain
Mississippi goes to Obama
Missouri goes to McCain
Montana goes to Obama
Nebraska goes to McCain
Nevada goes to Obama
New Hampshire goes to McCain
New Jersey goes to Obama
New Mexico goes to McCain
New York goes to Obama
North Carolina goes to McCain
North Dakota goes to Obama
Ohio goes to McCain
Oklahoma goes to Obama
Oregon goes to McCain
Pennsylvania goes to Obama
Rhode Island goes to McCain
South Carolina goes to Obama
South Dakota goes to McCain
Tennessee goes to Obama
Texas goes to McCain
Utah goes to Obama
Vermont goes to McCain
Virginia goes to Obama
Washington goes to McCain
West Virginia goes to Obama
Wisconsin goes to McCain
Wyoming goes to Obama
Electoral College vote total for John McCain (R) is 246
Electoral College vote total for Barack Obama (D) is 292
The next President of the United States is Barack Obama
```

zones01.c — zones05.c

```
> zones01
Enter X:3
Enter Y:5
blue
> zones01
Enter X:10
Enter Y:3
green
> zones02
Enter X:4
Enter Y:6
purple
> zones02
Enter X:6
Enter Y:4
blue
> zones02
Enter X:8
Enter Y:7
orange
> zones03
Enter X:0
Enter Y:2
red
> zones03
Enter X:3
Enter Y:7
> zones03
Enter X:7
Enter Y:5
orange
> zones03
Enter X:10
Enter Y:1
purple
> zones04
Enter X:0
Enter Y:1
red
> zones04
Enter X:6
Enter Y:6
blue
> zones04
Enter X:5
Enter Y:6
blue
> zones04
Enter X:5
Enter Y:5
red
> zones05
Enter X:1
Enter Y:4
red
> zones05
Enter X:4
Enter Y:1
> zones05
Enter X:6
Enter Y:7
red
> zones05
Enter X:7
Enter Y:6
```

Appendix B. Electoral College

No.	State	Abbr.	Population	Electoral College Votes
1.	Alabama	AL	4,500,752	**9**
2.	Alaska	AK	648,818	**3**
3.	Arizona	AZ	5,580,811	**10**
4.	Arkansas	AR	2,725,714	**6**
5.	California	CA	35,484,453	**55**
6.	Colorado	CO	4,550,688	**9**
7.	Connecticut	CT	3,483,372	**7**
8.	Delaware	DE	817,491	**3**
9.	District of Columbia	DC	563,384	**3**
10.	Florida	FL	17,019,068	**27**
11.	Georgia	GA	8,684,715	**15**
12.	Hawaii	HI	1,257,608	**4**
13.	Idaho	ID	1,366,332	**4**
14.	Illinois	IL	12,653,544	**21**
15.	Indiana	IN	6,195,643	**11**
16.	Iowa	IA	2,944,062	**7**
17.	Kansas	KS	2,723,507	**6**
18.	Kentucky	KY	4,117,827	**8**
19.	Louisiana	LA	4,496,334	**9**
20.	Maine	ME	1,305,728	**4**
21.	Maryland	MD	5,508,909	**10**
22.	Massachusetts	MA	6,433,422	**12**
23.	Michigan	MI	10,079,985	**17**
24.	Minnesota	MN	5,059,375	**10**
25.	Mississippi	MS	2,881,281	**6**
26.	Missouri	MO	5,704,484	**11**
27.	Montana	MT	917,621	**3**
28.	Nebraska	NE	1,739,291	**5**
29.	Nevada	NV	2,241,154	**5**
30.	New Hampshire	NH	1,287,687	**4**
31.	New Jersey	NJ	8,638,396	**15**
32.	New Mexico	NM	1,874,614	**5**
33.	New York	NY	19,190,115	**31**
34.	North Carolina	NC	8,407,248	**15**
35.	North Dakota	ND	633,837	**3**
36.	Ohio	OH	11,435,798	**20**
37.	Oklahoma	OK	3,511,532	**7**
38.	Oregon	OR	3,559,596	**7**
39.	Pennsylvania	PA	12,365,455	**21**
40.	Rhode Island	RI	1,076,164	**4**
41.	South Carolina	SC	4,147,152	**8**
42.	South Dakota	SD	764,309	**3**
43.	Tennessee	TN	5,841,748	**11**
44.	Texas	TX	22,118,509	**34**
45.	Utah	UT	2,351,467	**5**
46.	Vermont	VT	619,107	**3**
47.	Virginia	VA	7,386,330	**13**
48.	Washington	WA	6,131,445	**11**
49.	West Virginia	WV	1,810,354	**5**
50.	Wisconsin	WI	5,472,299	**10**
51.	Wyoming	WY	501,242	**3**