To many Americans, the federal government appears to be a "black box." Every year, on (or hopefully before) April 15, all law-abiding citizens write a check to the Internal Revenue Service to pay their taxes. In return, the government provides important services to its citizens: free public education, health care for the old and disabled, a guarantee of national security, and more. However, the details of this exchange are, for the most part, hidden from plain sight.

Politicians are constantly promising more services and fewer taxes for their constituents. If revenues decrease and expenditures increase, how does the government continue to pay for everything? The Treasury could mint extra money to give to the government, but as the amount of money increases, the value of money decreases, so this is generally a bad idea. Instead, the government borrows money from people, corporations, and other parts of the government. The amount that the government owes others is called debt.

You probably loaned some money to the government without realizing it. When you buy a Treasury bond, you are temporarily giving some of your own money to the government knowing that, when the bond matures, you will be paid interest. On the flip side, this means that the government has to pay interest on its debt every year. To pay this interest, the government borrows even more money, and the debt increases again. This sounds like a big problem, but it's not really. As long as people are willing to lend money to the government, the system works well.

Unfortunately, people who misunderstand how the economy operates perpetuate a myth that the total debt divided by the population is how much each citizen "owes." Combined with the current economic downturn, this means we are now in an environment where people are less willing to lend money to the government and are actually expecting the government to lend money to *them*!

It seems as though the obvious solution is to decrease the national debt by gradually raising income and lowering outcome. We endeavored to create a model of the federal budget that would keep track of spending and taxes to project how much the national debt will be in the future. Our simulation revealed some surprising facts about how potential policy changes in the next few years could drastically affect our nation's economy.

While the federal government may still be a black box, we have attempted to pry the cover open and shine a flashlight inside. Hopefully, what we have discovered will interest you as much as it has interested us.

National Debt and National Crisis

Team #2094

November 2008

Contents

1 Introduction

In the weeks leading up the 2008 U.S. presidential elections, the economy was in the throes of what one economist at New York University called "the worst financial crisis since the Great Depression" [1]. The economic situation quickly became one of the most important issues to voters. The president-elect, Barack Obama, will become responsible for a total national debt of over $10 trillion. During the debates preceding the election, Obama has advocated using a "scalpel" to reduce spending for specific programs [2]. Obama will need to use mathematical models to evaluate potential changes to the budget in terms of their impact on the national debt. This model aims to put the national debt in perspective by modeling it as the sum of its component parts: income and expenditures.

2 Background

The total United States Public Debt, commonly known as the National Debt, is the sum of the budget deficits. A budget deficit is defined as the amount by which government expenditures exceed government revenues in a given year. National debt can be categorized into two distinct categories: debt held by the public and intragovernmental debt. Debt held by the public is the total accumulation of the deficits (minus the surpluses) the federal government has incurred through time. It represents the total amount of money owed to the holders of U.S. securities, the method by which the government finances its debt. U.S. securities include treasury bills, treasury notes, treasury bonds, and U.S. saving bonds. Intragovernmental debt is debt held by the government itself. Wars, recessions, and tax cuts are major sources of large budget deficits and therefore of the debt. It is not surprising then that the debt has grown significantly in the most recent years with the wars in Iraq and Afghanistan, tax cuts, and the impending recession. The debt has grown from $5,674 billion in 2000, to $7,933 billion in 2005, to $10,024.7 in 2008 [3]. Since the dollar value of the debt itself does not carry much information, debt as a percentage of gross domestic product (GDP) is a good indicator of the impact of the national debt. The larger the percentage, the larger the debt is relative to the economy. The debt needs to be managed appropriately for long term stability.

2.1 Model Considerations

The yearly budget, a highly complex document, is broken down at a macro level into expenditures and revenue. The government is financed mainly through a variety of taxes. Expenditures include entitlements such as Social Security, Medicare, and Medicaid, discretionary funding such as government programs and operating budgets, and mandatory yearly interest on the national debt.

2.1.1 Funding the Government (Taxes)

The government is primarily funded through a variety of taxes including excise taxes (a tax changed on goods produced inside the coutry), corporate taxes (a tax levied on the profit made by a company), trade tariffs (customs duties), payroll taxes (federal income tax withholding, the Social Security tax, and the Medicare tax), estate taxes (a tax on the transfer of the "taxable estate" of a deceased person), and income taxes (a tax on the financial income of a person).

2.1.2 Entitlements

Social Security, officially referred to as the Old-Age, Survivors, and Disability Insurance (OASDI) program, is currently the largest entitlement program in the U.S. Budget. Social Security, a "pay-as-you-go" plan is funded through dedicated payroll taxes called Federal Insurance Contributions Act (FICA). The mandated benefits paid out each year are financed by the payroll tax revenues received each year. Every year since 1982, OASDI tax receipts, interest payments and other income have exceeded benefit payments and other expenditures [5]. As the "baby boomers" move out of the work force and into retirement, however, it is anticipated that expenses will come to exceed Social Security tax revenues if there are no changes in current laws concerning taxes, benefits, and the retirement age. The Social Security Admistration predicts that the trust fund built up on revenues from years of surpluses will be exhausted by 2041. Other Social Security programs include Medicare, a health insurance program for the aged and disabled, and Medicaid, a health program for needy individuals and families. Entitlements are dictated by the number of people who sign up for these benefits, rather than by Congress.

2.1.3 Discretionary Funding

Discretionary funding is funding that Congress explicitly determines on an annual basis. Discretionary funding includes money for government operations and its programs in addition to the military.

2.1.4 Servicing the Debt

The federal government must pay interest each year to service its debt. For instance, in 2003, $318 billion was spent on interest payments servicing the debt. As the debt increases, the the government is required to spend even more on its interest payments.

3 Model Design

We decided to construct a time-driven model due to all relevant factors being either time-based (e.g. GDP for a fiscal year) or continuous (e.g. population). The various factors of our model are based on a combination of trends found in

historical data and basic rules about how they should work (e.g. income taxes are calculated based on tax brackets and income distribution).

3.1 Population

The total size of the United States population and its age distribution are very important parts of our model. We model the population based on basic rules of population dynamics, starting from the 2006 population. Every year, the age distribution of the population is shifted upward by one year to account for aging. Then, newborns are added to the youngest age group based on birth rates for mothers in various age groups, assuming women make up 50% of every age group [6]. Finally, the population in each age group is reduced according to death rates for the different age groups [7]. The initial age group data is taken from the U.S. Census estimates for 2006, and uniformity is assumed in each age group except for the 0...4 age group, where the death rate is used to back-calculate the population in each year of age, in 2006 [8].

3.2 GDP

We assumed that GDP per capita (adjusted for inflation) is linear, GDP per capita (adjusted for inflation) is constant during recession, there's a recession 2008 and 2009 (justification: graph of historical GDP per capita), and inflation is constant.

3.3 Taxes

Our model considered a variety of different types of taxes to account for the government's annual revenue.

3.3.1 Excise Taxes

Office of Management and Budget (OMB) historical statistics and projections [9] indicated that excise taxes have remained and likely will remain constant. Since there was no clear trend among annual revenue from excise taxes and the spread was around 10 billion dollars (very small relative to the entire budget), we adjusted for inflation (4.28% annually) and found an average excise tax revenue. This constant (adjusted for inflation) was used as our 2009-2017 projections for revenue gained from excise taxes.

3.3.2 Corporate Taxes

When table S-9 of the OMB's FY2009 budget [9] is combined with our GDP projections, it becomes apparent that GDP is, on average, 41.07057738 times the total revenue from corporate taxes. We used this relationship and our GDP projection to calculate the yearly revenue from corporate taxes.

3.3.3 Trade Tariffs

It appears that the trade tariffs, when adjusted to 2007 dollars (assuming 4.28% annual inflation) were largely linear. Fitting the data to a linear regression (where x is the number of years since 2008 and y is in billions of 2007 dollars) resulted in the following line for which $R^2 = 0.973991997$: $y = .4695950714x + 22.57457906$.

3.3.4 Income Taxes

Adjusted gross income (AGI) is the income that is taxed by the income tax. We assumed that AGI varies directly with GDP. In order to predict income taxes, we also assumed a constant income distribution and that although tax brackets will be adjusted year to year, they will also float with increased GDP per capita so that nth percentile will always be in the same tax bracket. We also considered Obama's tax plan. It it, the top two tax brackets would return to to 1990s level and cause the return of effective tax rates for top 5% highest-earners to the 1990s level. We consider the top two tax brackets the two 5% of earners as they almost perfectly correspond based on income figures.

3.3.5 Payroll Taxes

We modeled three types of payroll taxes; Social Security tax: 6.2% on the wage base with employer matching, Medicare tax: 1.45% on all income with employer matching, Federal Unemployment Tax: although normally 0.8% on the first $7,000, will assume that it is always 0.8%. Using a household income distribution and assuming that the income distribution does not change, we calculated per capita payroll taxes for each group to find the total payroll taxes.

3.3.6 Estate and Gift Taxes

Under President Bush's repeal of estate taxes, they will be eventually phased out shortly after 2013 [9]. Using our GDP projections, we calculate that GDP divided by estate/gift taxes follows the regression $0.59957711903826(2.3294555089822)^x$ where x is the number of years since 2000. President-elect Obama has indicated that he plans to reinstate estate taxes to pre-repeal levels. If they were repealed, we assume they would return to the ratio of GDP to estate taxes in 2007 (534.53).

3.3.7 Other Taxes

There are other minor taxes accounted for in the budget. A simple linear regression where x=years since 2000 models this small amount of taxes. $-9.5689564762651x + 386.69650817756$

3.4 Expenditures

Expenditures are the way the government categorizes the money it spends.

3.4.1 Entitlements

Social Security programs in the United States include Federal Old-Age, Survivors, and Disability Insurance, Health Insurance for Aged and Disabled (Medicare), Grants to States for Medical Assistance Programs (Medicaid), and State Children's Health Insurance Program (SCHIP). In order to calculate the number of people receiving various types of Social Security benefits, we assume that the proportion of people above the retirement age receiving retirement benefits remains constant, as do the proportion of adults below retirement age receiving disability benefits and the proportion of people in each age group (0-18, 18-64, 65+) receiving SSI benefits. We used historical regressions of data to determine the average monthly payouts per person enrolled, which were [12, 13]:

- Retired workers: $.3125441165x^2 + 31.70485733x + 823.8547063$ (with $R^2 = .9968147621$)
- Non-disabled widow(er)s: $.3107754754x^2 + 31.33249266x + 800.347688$ (with $R^2 = .9971714305$)
- Disabled workers: $.2046799838x^2 + 26.5484932x + 788.2471737$ (with $R^2 = .9954809785$)

Similarly, for Medicaid and SCHIP programs, OMB projections [9] indicated that SCHIP/Medicaid spending will increase very linearly over the next six years. x is the number of years since 2000 and $R^2 = .9911769076$. $y = (16.5x + 77.85714286) \times 10^9$

3.4.2 Discretionary Spending

Discretionary spending is composed of "standard" discretionary spending and security discretionary spending. Non security discretionary spending is the most broad, including budgets for government agencies, the mililtary, and governmental programs. Security discretionary spending is a small category including money for special Homeland Security and Department of Defense projects. For both types of spending we used 2008 figures and adjusted for GDP.

3.4.3 Interest on the Debt

For each year, from 1962-2007, we calculated the "interest rate" as interest paid divided by total debt [14]. Next, for each year, we found the GDP divided by the interest rate GDP/rate (in \$billions/%). It appears to follow a cubic regression where x is the number of years since 1961. $R^2 = 0.968$ for the regression $y = 16.169x^3 - 671.34x^2 + 9075.2x - 83.162$

4 Model Testing and Sensitivity Analysis

There are a total of 432 possible states for our simulation. They include:

- Variable Retirement Age: 66, 68, 70 (It is possible that by increasing the retirement age the deficit and the national debt could be decreased)
- Medicare cap: true/false (A medicare cap, like the one President Bush has proposed would reduce costs as it would reduce coverage for the more expensive treatments. However, there would be significant public opposition to any such cap.)
- Repeal estate taxes: true/false (Estate taxes are currently being slowly phased out. President-elect Obama has indicated that he wants to reimplement the estate tax).
- Tax Plan: Bush, Obama, Custom Tax Plan * War Plan: Surge, Status Quo, Obama withdrawal, Afghanistan withdrawal
- Variable Inflation: 1.0428 (meaning 4.28% annually; the actual value for the U.S. economy), 1.0328, and 1.0528

Since income taxes are the most important, we focused on the changes Obama's proposed tax plan would bring [15]. The top two income tax brackets would return to the 1990s levels of 36% and 39.6%. All other tax brackets would remain as they are today. According to the tax code, the top two brackets correspond to $164,551 and above in 2008, meaning these Obama effects will be felt by people earning over $164,000. These people roughly correspond with the top 5% of the population, since according to the US Census Bureau in 2006 [16] the top 5% earns $167,000 and up. (Since these earnings have obviously increased in 2008, this goes over $164,551 a bit, but this is fine, since not all income is taxed, which corrects for the error.) Thus, the average tax for the top 5% of people was made to return to 1990's levels to reflect Obama's income tax hike. The average 1990s level for the top 5% was around 24% according to [11], so that is all that was changed in the tax array to account for Obama's plans, since he stated other tax brackets would remain the same. A limitation of this approach becomes evident: not all income tax for people who earn in the top bracket is calculated using the top bracket; other brackets come into play too.

We also considered different war plans The Department of Defense is currently spending is $479.5 billion [9]. The war in Afghanistan began in 2001, and the war in Iraq began in 2003. An Iraq withdrawal would return DoD funding to 2002 levels, and an Iraq/Afghanistan withdrawal would return DoD funding to 2000 levels (plus regular inflation per year) [13].

5 References

1. http://online.wsj.com/article/SB122169431617549947.html
2. http://www.marketwatch.com/news/story/mccain-proposes-spending-freeze-obama/story.aspx?guid={EA5E114C-A041-42A1-A869-D61FFC1DF874}&-siteid=rss
3. http://www.treasurydirect.gov/govt/reports/pd/histdebt/histdebt_histo5.htm
4. http://www.law.cornell.edu/uscode/html/uscode42/usc_sec_42_00000401—-000-.html
5. http://www.ssa.gov/OACT/STATS/table4a3.html
6. ftp://ftp.cdc.gov/pub/Health_Statistics/NCHS/Dataset_Documentation/-DVS/natality/UserGuide2005.pdf
7. http://www.disastercenter.com/cdc/Death%20rates%202005.html
8. http://www.census.gov/popest/national/asrh/NC-EST2006/NC-EST2006-01.xls
9. http://www.whitehouse.gov/omb/budget/fy2009/summarytables.html
10. https://www.cia.gov/library/publications/the-world-factbook/geos/us.html
11. http://www.taxfoundation.org/research/show/250.html
12. http://www.infoplease.com/ipa/A0780010.html
13. http://www.ssa.gov/policy/docs/quickfacts/stat_snapshot
14. http://www.whitehouse.gov/omb/budget/fy2009/pdf/hist.pdf
15. http://www.barackobama.com/pdf/taxes/Factsheet_Tax_Plan_FINAL.pdf
16. http://en.wikipedia.org/wiki/Income_in_the_United_States

```
public class DebtSimulation extends TimeSimulation<DebtState>
{
    public int RETIREMENT_AGE = 66;
    public boolean CAP_MEDICARE_FOR_INFLATION = true;
    public boolean REPEAL_ESTATE_TAXES = false;
    public TaxPlan TAX_PLAN = TaxPlan.CHANGE;
    public WarPlan WAR_PLAN = WarPlan.OBAMA_WITHDRAWAL;
    public double INFLATION = 1.0428;

    public boolean DEBUG = false;

    Budget budget;
    Population population;
    public static final double GDP_2008 = 14334e9;

    public DebtSimulation(double starttime, double endtime, double dt)
    {
        super(starttime, endtime, dt);
    }

    public static void main(String[] main)
    {
        DebtSimulation db = new DebtSimulation(2009, 2017, 1);
        db.run();
        db.showStats();
    }

    public void initialize()
    {
        budget = new Budget();
        population = new Population();
        current = new DebtState();
        current.setDebt(10024700000000.0);
        current.setGDP(GDP_2008);
        next = new DebtState();
    }

    public void update()
    {
        /*
         * A number of factors contribute to the yearly deficit. They include
         * those that reduce deficit / add to a surplus: corporate taxes, social
         * security taxes, constitutional revenues such as excise taxes on
         * cigarettes, alcohol, tobacco, firearms, tires, etc., and tariffs on
         * trade.
         */
        if (DEBUG)
            System.out.println("-----YEAR " + intTime + "-----");

        next.setDebt(current.getDebt() + budget.getDeficit(this));
        //Assuming GDP per capita is linear and faces
        //  a 4.28% inflation rate
        next.setGDP((current.getGDP() / population.getTotalNumberOfPeople(intTime - 1) +
 715) * population.getTotalNumberOfPeople(intTime) * INFLATION);

        if (DEBUG)
        {
            System.out.println("National Debt:\t\t$" + (int) (next.getDebt() /
 Math.pow(10, 9)) + " billion");
            System.out.println("Population:\t\t" +
 population.getTotalNumberOfPeople(intTime));
            System.out.println("Per Capita Debt:\t$" + (int) (next.getDebt() /
 population.getTotalNumberOfPeople(intTime)));
```

```
            System.out.println("GDP:\t\t\t$" + (int) (next.getGDP() / Math.pow(10, 9)) +
 " billion");
            System.out.println("Debt/GDP:\t\t" + (int) (next.getDebt() / next.getGDP() *
 1000) / 10.0 + "%\n");
        }
    }

    /**
     * Calculate Statistics
     */
    public void finish()
    {
        // TODO finish()
    }

    public void showStats()
    {
        System.out.println("Settings:");
        System.out.println("\tRetirement age:\t" + RETIREMENT_AGE);
        System.out.println("\tCap Medicare?:\t" + (CAP_MEDICARE_FOR_INFLATION ? "yes" :
 "no"));
        System.out.println("\tEstate taxes:\t" + (REPEAL_ESTATE_TAXES ? "repeal" :
 "keep"));
        System.out.println("\tTax plan:\t" + TAX_PLAN);
        System.out.println("\tWar plan:\t" + WAR_PLAN);
        System.out.println("\tInflation:\t" + (int) Math.round((INFLATION - 1.0) * 10000)
 / 100.0 + "%");
        System.out.println("National Debt in " + (int) this.endtime + ":\t$" + (int)
 (current.getDebt() / Math.pow(10, 9)) + " billion");
        System.out.println("Per Capita Debt:\t$" + (int) (current.getDebt() /
 population.getTotalNumberOfPeople((int) Math.round(endtime))));
        System.out.println("Debt/GDP:\t\t" + (int) (current.getDebt() / current.getGDP() *
 1000) / 10.0 + "%");
    }
}
```

```java
public class DebtState extends State
{
    private double debt;
    private double GDP;

    public double getDebt()
    {
        return debt;
    }

    public void setDebt(double d)
    {
        debt = d;
    }

    public void setGDP(double GDP)
    {
        this.GDP = GDP;
    }

    public double getGDP()
    {
        return GDP;
    }
}
```

```java
public class Budget
{
    /**
     * If there is a deficit, the return is positive. If there is a surplus, the
     * return is negative.
     */
    public double getDeficit(DebtSimulation sim)
    {
        double deficit = 0.0;
        double revenues = 0.0;
        double expenditures = 0.0;

        if (sim.DEBUG)
            System.out.println("   --REVENUES--");
        // These reduce the deficit
        for (Revenue revenue : Revenue.values())
            revenues += revenue.getRevenue(sim);

        if (sim.DEBUG)
            System.out.println("   --EXPENDITURES--");
        // These increase the deficit
        for (Expenditure expenditure : Expenditure.values())
            expenditures += expenditure.getExpenditure(sim);

        deficit = expenditures - revenues;

        if (sim.DEBUG)
        {
            System.out.println("   --" + sim.intTime + " SUMMARY--");
            System.out.println("Revenue:\t\t$" + (int) (revenues / Math.pow(10, 9)) + "
billion");
            System.out.println("Expenditures:\t\t$" + (int) (expenditures / Math.pow(10,
 9)) + " billion");
            System.out.println("Deficit:\t\t$" + (int) (deficit / Math.pow(10, 9)) + "
billion");
        }

        return deficit;
    }
}
```

```
public enum Expenditure
{
    SOCIAL_SECURITY
    {
        public double getExpenditure(DebtSimulation sim)
        {
            double totalExpenditure = 0.0;

            //based on regression of data from http://www.infoplease.com/ipa/A0780010.html
            //  and tables 2 and 3 of
http://www.ssa.gov/policy/docs/quickfacts/stat_snapshot/
            //  we determined that the average monthly
            //  Social Security benefits are (x=years since 2000):
            //    *retired workers: .3125441165x^2+31.70485733x+823.8547063
(R^2=.9968147621)
            //    *non-disabled widow(er)s: .3107754754x^2+31.33249266x+800.347688
(R^2=.9971714305)
            //    *disabled workers: .2046799838x^2+26.5484932x+788.2471737
(R^2=.9954809785)
            //  Assumptions:
            //    *retirement age is 66 (about average for people
            //       retiring around now, as per
http://www.ssa.gov/OACT/ProgData/nra.html)
            //    *everyone retires at the retirement age and starts
            //       receiving Social Security benefits then
            //    *recipients of disability/widow(er) benefits are
            //       between 18 and 65 years old
            //    *average monthly SSI payments are as given
            //       in table 3 of the site referenced above
            //       and will not change over time, except for
            //       regular inflation
            //    *the following percentages will remain constant
            //       (each is calculated based on the data given
            //       in tables 2 and 3 and our projections for
            //       2008 populations):
            //       -percentage of those (66 and older) receiving
            //          Social Security benefits
            //       -percentage of those (65 and younger) receiving
            //          Social Security disability or widow(er) benefits
            //       -percentage of those in three age groups (under 18,
            //          18-64, 65 and older) receiving SSI benefits

            int receivingBenefitsIn2008;
            int populationSizeIn2008;
            int projectedPopulationSize;
            double projectedAverageMonthlyPayment;
            double projectedTotalPayments;

            //Social Security old age payments
            receivingBenefitsIn2008 = 1000 * (32149 + 2392 + 503);
            populationSizeIn2008 = sim.population.getNumberOfPeople(2008,
 sim.RETIREMENT_AGE, 85);
            projectedPopulationSize = sim.population.getNumberOfPeople(sim.intTime,
 sim.RETIREMENT_AGE, 85);
            projectedAverageMonthlyPayment = .3125441165 * (sim.time - 2000) * (sim.time -
 2000) + 31.70485733 * (sim.time - 2000) + 823.8547063;
            projectedTotalPayments = 12.0 * ((double) receivingBenefitsIn2008 /
 populationSizeIn2008) * projectedPopulationSize * projectedAverageMonthlyPayment;
            totalExpenditure += projectedTotalPayments;

            //Social Security disability payments
            receivingBenefitsIn2008 = 1000 * (7300 + 150 + 1655);
            populationSizeIn2008 = sim.population.getNumberOfPeople(2008, 18,
```

```
sim.RETIREMENT_AGE - 1);
            projectedPopulationSize = sim.population.getNumberOfPeople(sim.intTime, 18,
sim.RETIREMENT_AGE - 1);
            projectedAverageMonthlyPayment = .2046799838 * (sim.time - 2000) * (sim.time -
2000) + 26.5484932 * (sim.time - 2000) + 788.2471737;
            projectedTotalPayments = 12.0 * ((double) receivingBenefitsIn2008 /
populationSizeIn2008) * projectedPopulationSize * projectedAverageMonthlyPayment;
            totalExpenditure += projectedTotalPayments;

            //Social Security widow(er) payments
            receivingBenefitsIn2008 = 1000 * (4399 + 159 + 1862);
            populationSizeIn2008 = sim.population.getNumberOfPeople(2008, 18,
sim.RETIREMENT_AGE - 1);
            projectedPopulationSize = sim.population.getNumberOfPeople(sim.intTime, 18,
sim.RETIREMENT_AGE - 1);
            projectedAverageMonthlyPayment = .3107754754 * (sim.time - 2000) * (sim.time -
2000) + 31.33249266 * (sim.time - 2000) + 800.347688;
            projectedTotalPayments = 12.0 * ((double) receivingBenefitsIn2008 /
populationSizeIn2008) * projectedPopulationSize * projectedAverageMonthlyPayment;
            totalExpenditure += projectedTotalPayments;

            //SSI payments (under 18)
            receivingBenefitsIn2008 = 1000 * 1148;
            populationSizeIn2008 = sim.population.getNumberOfPeople(2008, 0, 17);
            projectedPopulationSize = sim.population.getNumberOfPeople(sim.intTime, 0,
17);
            projectedAverageMonthlyPayment = 566.00 * Math.pow(sim.INFLATION, sim.time -
2008);
            projectedTotalPayments = 12.0 * ((double) receivingBenefitsIn2008 /
populationSizeIn2008) * projectedPopulationSize * projectedAverageMonthlyPayment;
            totalExpenditure += projectedTotalPayments;

            //SSI payments (18-64)
            receivingBenefitsIn2008 = 1000 * 4329;
            populationSizeIn2008 = sim.population.getNumberOfPeople(2008, 18, 64);
            projectedPopulationSize = sim.population.getNumberOfPeople(sim.intTime, 18,
64);
            projectedAverageMonthlyPayment = 491.90 * Math.pow(sim.INFLATION, sim.time -
2008);
            projectedTotalPayments = 12.0 * ((double) receivingBenefitsIn2008 /
populationSizeIn2008) * projectedPopulationSize * projectedAverageMonthlyPayment;
            totalExpenditure += projectedTotalPayments;

            //SSI payments (65 or older)
            receivingBenefitsIn2008 = 1000 * 2033;
            populationSizeIn2008 = sim.population.getNumberOfPeople(2008, 65, 85);
            projectedPopulationSize = sim.population.getNumberOfPeople(sim.intTime, 65,
85);
            projectedAverageMonthlyPayment = 394.10 * Math.pow(sim.INFLATION, sim.time -
2008);
            projectedTotalPayments = 12.0 * ((double) receivingBenefitsIn2008 /
populationSizeIn2008) * projectedPopulationSize * projectedAverageMonthlyPayment;
            totalExpenditure += projectedTotalPayments;

            if (sim.DEBUG)
                System.out.println("Social Security/SSI:\t$" + (int) (totalExpenditure /
Math.pow(10, 9)) + " billion");

            return totalExpenditure;
        }
    },
    MEDICARE
    {
```

```
        public double getExpenditure(DebtSimulation sim)
        {
            double totalExpenditure = 0.0;

            //Medicare spending per enrollee historical
            //  data from http://statehealthfacts.org/comparemaptable.jsp?ind=624&cat=6
            //Regression on the data gives that per enrollee
            //  spending is (where x=years since 2000):
            //  3.5734265734271x^3+35.466200466202x^2+238.35664335664x+5737.8834498835
(R^2=.9795459611)
            //http://statehealthfacts.org/comparebar.jsp?ind=294&cat=6
            //  shows the percentage of Medicare recipients
            //  in various age groups in 2007
            //  *ASSUMING that these percentages stay constant,
            //   it can be calculated from the 2008 total enrollment
            //   (http://statehealthfacts.org/comparemaptable.jsp?ind=290&cat=6)
            //   how many beneficiaries there are in each age group
            //  *next, using the population model for 2008,
            //   the percentage of the total population in each
            //   age group can be calculated
            //  *then ASSUMING these percentages also stay constant,
            //   the number of beneficiaries in any year can be
            //   found from the population model
            //  *multiplied by the projected per enrollee cost,
            //   this gives the total cost for Medicare

            double projectedPerEnrolleeCost = 3.5734265734271 * (sim.time - 2000) *
(sim.time - 2000) * (sim.time - 2000) + 35.466200466202 * (sim.time - 2000) * (sim.time -
2000) + 238.35664335664 * (sim.time - 2000) + 5737.8834498835;

            //ALTERNATIVE MODEL:
            //  We can say that, since 2004, the per enrollee
            //  cost will increase only for inflation

            if (sim.CAP_MEDICARE_FOR_INFLATION)
                projectedPerEnrolleeCost = 7439 * Math.pow(sim.INFLATION, sim.time -
2004);

            double numberOfBeneficiariesIn2008;
            double percentReceivingBenefits;
            double numberOfBeneficiaries;

            //adults 19-64
            numberOfBeneficiariesIn2008 = 44831390.0 * .160;
            percentReceivingBenefits = numberOfBeneficiariesIn2008 /
sim.population.getNumberOfPeople(2008, 19, 64);
            numberOfBeneficiaries = percentReceivingBenefits *
sim.population.getNumberOfPeople(sim.intTime, 19, 64);
            totalExpenditure += numberOfBeneficiaries * projectedPerEnrolleeCost;

            //elderly 65-74
            numberOfBeneficiariesIn2008 = 44831390.0 * .430;
            percentReceivingBenefits = numberOfBeneficiariesIn2008 /
sim.population.getNumberOfPeople(2008, 65, 74);
            numberOfBeneficiaries = percentReceivingBenefits *
sim.population.getNumberOfPeople(sim.intTime, 65, 74);
            totalExpenditure += numberOfBeneficiaries * projectedPerEnrolleeCost;

            //elderly 75-84
            numberOfBeneficiariesIn2008 = 44831390.0 * .301;
            percentReceivingBenefits = numberOfBeneficiariesIn2008 /
sim.population.getNumberOfPeople(2008, 75, 84);
            numberOfBeneficiaries = percentReceivingBenefits *
```

```
 sim.population.getNumberOfPeople(sim.intTime, 75, 84);
            totalExpenditure += numberOfBeneficiaries * projectedPerEnrolleeCost;

            //elderly 85+
            numberOfBeneficiariesIn2008 = 44831390.0 * .100;
            percentReceivingBenefits = numberOfBeneficiariesIn2008 /
 sim.population.getNumberOfPeople(2008, 85, 85);
            numberOfBeneficiaries = percentReceivingBenefits *
 sim.population.getNumberOfPeople(sim.intTime, 85, 85);
            totalExpenditure += numberOfBeneficiaries * projectedPerEnrolleeCost;

            if (sim.DEBUG)
                System.out.println("Medicare:\t\t$" + (int) (totalExpenditure /
 Math.pow(10, 9)) + " billion");

            return totalExpenditure;
        }
    },
    MEDICAID_AND_SCHIP
    {
        public double getExpenditure(DebtSimulation sim)
        {
            double totalExpenditure = 0.0;

            //OMB projections (table S-8 at
http://www.whitehouse.gov/omb/budget/fy2009/summarytables.html)
            //  show that SCHIP/Medicaid spending will increase
            //  very linearly over the next six years, with
            //  regression showing (x=years since 2000):
            //  16.5x+77.85714286 in $billion (R^2=.9911769076)

            totalExpenditure = (16.5 * (sim.time - 2000) + 77.85714286) * 1000000000;

            if (sim.DEBUG)
                System.out.println("Medicaid/SCHIP:\t\t$" + (int) (totalExpenditure /
 Math.pow(10, 9)) + " billion");

            return totalExpenditure;
        }
    },
    OTHER_ENTITLEMENT
    {
        public double getExpenditure(DebtSimulation sim)
        {
            double totalExpenditure = 0.0;

            //OMB projections (table S-8 at
http://www.whitehouse.gov/omb/budget/fy2009/summarytables.html)
            //  show that other mandatory spending is about
            //  quadratic over the next six years, with
            //  regression showing (x=years since 2000):
            //  -2.714285714x^2+68.07142857x-37.28571429
            //    (in $billion) with R^2=0.9610548699

            totalExpenditure = (-2.714285714 * (sim.time - 2000) * (sim.time - 2000) +
 68.07142857 * (sim.time - 2000) - 37.28571429) * 1000000000;

            if (sim.DEBUG)
                System.out.println("Other entitlement:\t$" + (int) (totalExpenditure /
 Math.pow(10, 9)) + " billion");

            return totalExpenditure;
        }
```

```
    },
    NONSECURITY_DISCRETIONARY_SPENDING
    {
        public double getExpenditure(DebtSimulation sim)
        {
            double[] departments = { //data from table S-3 at
http://www.whitehouse.gov/omb/budget/fy2009/summarytables.html
                    /* agriculture */21.8,
                    /* commerce */6.9,
                    /* defense */sim.WAR_PLAN.getSpending(sim),
                    /* education */57.2,
                    /* energy */23.9,
                    /* healthHumanServices */71.9,
                    /* homelandSecurity */34.9,
                    /* housingUrbanDevelopment */37.4,
                    /* interior */11,
                    /* justice */22.7,
                    /* labor */11.4,
                    /* statePrograms */32.9,
                    /* transportation */15.5,
                    /* treasury */12,
                    /* veteransAffairs */39.4,
                    /* corpsofEngineers */5.6,
                    /* environmentalProtectionAgency */7.5,
                    /* executiveOfficePresident */0.3,
                    /* judicialBranch */5.8,
                    /* legislativeBranch */4,
                    /* nationalAeronauticsSpaceAdministration */17.1,
                    /* nationalScienceFoundation */6,
                    /* smallBusinessAdministration */0.6,
                    /* socialSecurityAdministration */8,
                    /* otherAgencies */8 };

            double totalExpenditure = 0.0;
            for (double department : departments)
                totalExpenditure += 1e9 * department;
            totalExpenditure *= sim.current.getGDP() / DebtSimulation.GDP_2008;

            if (sim.DEBUG)
                System.out.println("Disc. (non-security):\t$" + (int) (totalExpenditure /
 Math.pow(10, 9)) + " billion");

            return totalExpenditure;
        }
    },
    SECURITY_DISCRETIONARY_SPENDING
    {
        public double getExpenditure(DebtSimulation sim)
        {
            double[] departments = { //data from table S-4 at
http://www.whitehouse.gov/omb/budget/fy2009/summarytables.html
                    /* agriculture */571,
                    /* commerce */207,
                    /* defense */17375,
                    /* energy */1830,
                    /* healthHumanServices */4300,
                    /* homelandSecurity */30093,
                    /* justice */3273,
                    /* state */1962,
                    /* transportation */206,
                    /* treasury */117,
                    /* veteransAffairs */271,
                    /* environmentalProtectionAgency */138,
```

```
                    /* generalServicesAdministration */143,
                    /* nationalAeronauticsSpaceAdministration */205,
                    /* nationalScienceFoundation */374,
                    /* socialSecurityAdministration */212,
                    /* intelligenceCommunityManagement */122,
                    /* nuclearRegulatoryCommission */72,
                    /* smithsonianInstitution */93,
                    /* otherAgencies */236 };

            double totalExpenditure = 0.0;
            for (double department : departments)
                totalExpenditure += 1e6 * department;
            totalExpenditure *= sim.current.getGDP() / DebtSimulation.GDP_2008;

            if (sim.DEBUG)
                System.out.println("Disc. (security):\t$" + (int) (totalExpenditure /
 Math.pow(10, 9)) + " billion");

            return totalExpenditure;
        }
    },
    DEBT_INTEREST
    {
        public double getExpenditure(DebtSimulation sim)
        {
            double totalExpenditure = 0.0;

            //based on the following data from
http://www.whitehouse.gov/omb/budget/fy2009/pdf/hist.pdf
            //  *net interest (table 3.2)
            //  *GDP (table 6.1)
            //  *gross federal debt (table 7.1)
            //for each year, from 1962-2007, we calculated
            //  the "interest rate" as interest paid
            //  divided by total debt
            //next, for each year, we found the GDP
            //  divided by the interest rate
            //GDP/rate (in $billions/%) seems to follow a
            //  cubic regression (x=years since 1961):
            //  16.169*x^3 - 671.34*x^2 + 9075.2*x - 83.162 (R^2=0.968)

            double GDPperRate = 16.169 * (sim.time - 1961) * (sim.time - 1961) * (sim.time
 - 1961) - 671.34 * (sim.time - 1961) * (sim.time - 1961) + 9075.2 * (sim.time - 1961) -
 83.162;
            double interestRate = (sim.current.getGDP() / 1e9) / GDPperRate;
            double interest = interestRate * sim.current.getDebt();
            totalExpenditure = interest;

            if (sim.DEBUG)
                System.out.println("Debt interest:\t\t$" + (int) (totalExpenditure /
 Math.pow(10, 9)) + " billion");

            return totalExpenditure;
        }
    };

    public abstract double getExpenditure(DebtSimulation sim);
}
```

```
public enum Revenue
{
    EXCISE_TAX
    {
        /*
         * Source:
         * http://www.whitehouse.gov/omb/budget/fy2009/summarytables.html
         * Numbers were first adjusted for 4.28% inflation (to convert to 2007
         * dollars), then averaged (the spread was only ~10 billion), and this
         * average plus inflation is used for our projections.
         */
        public double getRevenue(DebtSimulation sim)
        {
            double totalRevenue = 59.1e9 * Math.pow(sim.INFLATION, sim.time - 2007); //in
USD

            if (sim.DEBUG)
                System.out.println("Excise Taxes:\t\t$" + (int) (totalRevenue /
 Math.pow(10, 9)) + " billion");

            return totalRevenue;
        }
    },
    CORPORATE_TAX
    {
        public double getRevenue(DebtSimulation sim)
        {
            double totalRevenue = 0.0;

            //based on table S-9 at
http://www.whitehouse.gov/omb/budget/fy2009/summarytables.html
            //  and our GDP projections, we calculated that
            //  corporate income taxes are, on average,
            //  1/41.07057738 of the total GDP

            totalRevenue = sim.current.getGDP() / 41.07057738;

            if (sim.DEBUG)
                System.out.println("Corporate Taxes:\t$" + (int) (totalRevenue /
 Math.pow(10, 9)) + " billion");

            return totalRevenue;
        }
    },
    TRADE_TARIFFS
    {
        /*
         * Source:
         * http://www.whitehouse.gov/omb/budget/fy2009/summarytables.html Data
         * was adjusted to 2007 dollars (4.28% inflation), 2008 was removed as
         * an outlier, and the data was fit to a linear regression:
         * y=.4695950714x+22.57457906 (R^2=0.973991997) where x=years since 2000
         * and y is in billions of 2007 dollars
         */
        public double getRevenue(DebtSimulation sim)
        {
            double totalRevenue = (.4695950714 * (sim.time - 2000) + 22.57457906) * 1e9 *
 Math.pow(sim.INFLATION, sim.time - 2007);

            if (sim.DEBUG)
                System.out.println("Trade Tariffs:\t\t$" + (int) (totalRevenue /
 Math.pow(10, 9)) + " billion");
```

```
            return totalRevenue;
        }
    },
    INCOME_TAX
    {
        /*
         * Source 1: http://www.taxfoundation.org/research/show/250.html AGI is
         * adjustable gross income, and it is the income that is taxed by the
         * income tax. We assumed that AGI varies directly with GDP. To find the
         * constant of variation, we used the CIA World Factbook to find the GDP
         * in 2006 and Source 1 to find the AGI in 2006. After dividing the two
         * numbers, we derived the formula AGI = GDP / 1.6166. The next step was
         * to see how the AGI was distributed across different income groups.
         * For simplicity, we assumed this income distribution is constant with
         * respect to time. This fixed AGI distribution was modeled by the share
         * array: share[0] = % of AGI held by the richest 5% of taxpayers
         * share[1] = " " " " " 6% - 10% group share[2] = " " " " " 11% - 25%
         * group share[3] = " " " " " 26% - 50% group share[4] = " " " " "
         * bottom 50% group To fill in the array, 2006 data from Source 1 was
         * used. Next we represented the tax policy by writing the average
         * percent tax for each income group. This is represented by the tax
         * array: for example, tax[1] holds the average tax rate for the 6% -
         * 10% group. To account for different tax polcies, merely change the
         * values in the tax array. Finally, the total income tax revenue is the
         * weighted average tax rate times the total AGI.
         */
        public double getRevenue(DebtSimulation sim)
        {
            double totalRevenue = 0.0;

            final double[] share = {
                    36.66, 10.66, 20.84, 19.33, 12.51 }; //assume constant income
distribution
            double AGI = sim.current.getGDP() / 1.6166; //Adjusted Gross Income
correlates with GDP

            for (int i = 0; i < share.length; i++)
                totalRevenue += (share[i] / 100) * AGI * (sim.TAX_PLAN.tax[i] / 100);

            if (sim.DEBUG)
                System.out.println("Income Taxes:\t\t$" + (int) (totalRevenue /
 Math.pow(10, 9)) + " billion");

            return totalRevenue;
        }
    },
    PAYROLL_TAXES
    {
        public double getRevenue(DebtSimulation sim)
        {
            double totalRevenue = 0.0;

            //data from http://en.wikipedia.org/wiki/Payroll_tax#United_States
            //Social Security tax: 6.2% on the wage base,
            //  employer matches that amount (wage base
            //  was $102,000 in 2008 and increases by
            //  4.1429843% on average each year)
            //Medicare tax: 1.45% on all income, employer
            //  matches that amount
            //FUTA tax: is normally 0.8% on the first $7,000,
            //  will assume that it is always 0.8%

            //data from http://en.wikipedia.org/wiki/Household_income_in_the_United_States
```

```
            //assuming that incomes are distributed
            //  uniformly within each range and that
            //  the mean income in the last range
            //  ($250,000 and more) is $500,000
            //all data is in 2005 dollars, need to adjust for inflation
            int[] dataRanges = {
                    1250, 3750, 6250, 8750, 11250, 13750, 16250, 18750, 21250,
                    23750, 26250, 28750, 31250, 33750, 36250, 38750, 41250,
                    43750, 46250, 48750, 51250, 53750, 56250, 58750, 61250,
                    63750, 66250, 68750, 71250, 73750, 76250, 78750, 81250,
                    83750, 86250, 88750, 91250, 93750, 96250, 98750, 125000,
                    175000, 225000, 500000 };
            int[] numberOfHouseholds = {
                    2566, 1389, 2490, 3360, 4013, 3543, 3760, 3438, 4061, 3375,
                    3938, 2889, 3921, 2727, 3360, 2633, 3378, 2294, 2700, 2371,
                    3071, 2006, 2420, 1786, 2566, 1774, 2101, 1637, 1978, 1413,
                    1802, 1264, 1673, 1219, 1418, 984, 1282, 917, 1023, 846,
                    11194, 3595, 1325, 1699 }; //in thousands
            double[] meanHouseholdSize = {
                    1.97, 2.04, 1.76, 1.66, 1.71, 1.84, 1.99, 2.1, 2.11, 2.5,
                    2.21, 2.3, 2.38, 2.39, 2.36, 2.49, 2.46, 2.6, 2.6, 2.62,
                    2.6, 2.72, 2.75, 2.87, 2.82, 2.89, 2.93, 2.8, 2.88, 3,
                    2.95, 3.04, 3.01, 3.1, 3, 3.03, 3.03, 3.25, 3.29, 3.33, 3,
                    3, 3, 3 };
            double[] meanNumberWorkers = {
                    0.23, 0.52, 0.39, 0.33, 0.46, 0.5, 0.67, 0.73, 0.84, 1,
                    0.93, 1.01, 1.12, 1.17, 1.22, 1.25, 1.31, 1.38, 1.39, 1.49,
                    1.46, 1.58, 1.61, 1.7, 1.63, 1.79, 1.81, 1.74, 1.77, 1.82,
                    1.82, 1.98, 1.89, 1.97, 1.94, 1.98, 1.95, 2.17, 2.06, 2.12,
                    2, 2, 2, 2 };

            //assume that, over time, the income distribution
            //  does NOT change, so that the number of people
            //  in each group increases by the same percentage
            //  that the total population increases
            //calculating the total number of people to find
            //  the scaling factor (i.e., the population growth)
            int totalNumberOfPeople = 0;
            for (int i = 0; i < dataRanges.length; i++)
                totalNumberOfPeople += numberOfHouseholds[i] * 1000 *
meanHouseholdSize[i];
            double populationScalingFactor = (double)
sim.population.getTotalNumberOfPeople(sim.intTime) / totalNumberOfPeople;

            //calculate the current Social Security wage base
            //based on historical data, the annual increase
            //  was calculated to be 4.1429843%
            double SocialSecurityBase = 102000;
            SocialSecurityBase *= Math.pow(1.041429843, sim.intTime - 2008);

            //for each income group, calculate the per capita
            //  payroll taxes to find the total payroll taxes
            //  paid by this group
            for (int i = 0; i < dataRanges.length; i++)
            {
                double incomePC = (double) dataRanges[i] * Math.pow(sim.INFLATION,
sim.time - 2005) / meanNumberWorkers[i]; //income per capita
                double FUTAPC = Math.min(0.008 * 7000, 0.008 * incomePC);
                double MedicarePC = incomePC * 0.029;
                double SocialSecurityPC = Math.min(0.124 * SocialSecurityBase, 0.124 *
incomePC);
                double payrollTaxesPC = FUTAPC + MedicarePC + SocialSecurityPC;
                double workers = numberOfHouseholds[i] * 1000 * meanNumberWorkers[i] *
```

```
 populationScalingFactor;
                totalRevenue += payrollTaxesPC * workers;
            }

            if (sim.DEBUG)
                System.out.println("Payroll Taxes:\t\t$" + (int) (totalRevenue /
 Math.pow(10, 9)) + " billion");

            return totalRevenue;
        }
    },
    ESTATE_AND_GIFT_TAXES
    {
        public double getRevenue(DebtSimulation sim)
        {
            double totalRevenue = 0.0;

            //based on table S-9 at
http://www.whitehouse.gov/omb/budget/fy2009/summarytables.html
            //  and our GDP projections, we calculated that
            //  GDP divided by estate/gift taxes follows the regression
            //  0.59957711903826*2.3294555089822^x (x=years since 2000)
            //note: currently, estate taxes are being repealed, but this ratio
            //  does not go to infinity because gift taxes are not being repealed

            double ratio = 0.59957711903826 * Math.pow(2.3294555089822, sim.time - 2000);

            //ALTERNATE PLAN:
            //  Obama opposes the repeal of estate taxes
            //  the ratio of GDP to estate taxes in 2007 was 534.53,
            //    so assume that the ratio would stay constant if
            //    estate taxes are not repealed

            if (!sim.REPEAL_ESTATE_TAXES)
                ratio = 534.53;

            totalRevenue = sim.current.getGDP() / ratio;

            if (sim.DEBUG)
                System.out.println("Estate/Gift Taxes:\t$" + (int) (totalRevenue /
 Math.pow(10, 9)) + " billion");

            return totalRevenue;
        }
    },
    OTHER_TAXES
    {
        public double getRevenue(DebtSimulation sim)
        {
            double totalRevenue = 0.0;

            //based on table S-9 at
http://www.whitehouse.gov/omb/budget/fy2009/summarytables.html
            //  and our GDP projections, we calculated that
            //  GDP divided by other taxes follows the regression
            //  -9.5689564762651x+386.69650817756 (x=years since 2000)
            //  (when the year 2007 is excluded as an outlier)

            double ratio = -9.5689564762651 * (sim.time - 2000) + 386.69650817756;
            totalRevenue = sim.current.getGDP() / ratio;

            if (sim.DEBUG)
                System.out.println("Other Taxes:\t\t$" + (int) (totalRevenue /
```

```
 Math.pow(10, 9)) + " billion");

            return totalRevenue;
        }
    };

    public abstract double getRevenue(DebtSimulation sim);
}
```

```java
public enum TaxPlan
{
    /*
     * (Documentation on how the tax array works is in CurrentRevenue.java under
     * INCOME_TAX.) Source 1:
     * http://www.taxfoundation.org/research/show/250.html BUSH'S TAX PLAN: Used
     * Source 1's data on average tax rates during 2006 (the most recent
     * available). OBAMA'S TAX PLAN: According to Obama's web site
     * http://www.barackobama.com/pdf/taxes/Factsheet_Tax_Plan_FINAL.pdf
     * "Ordinary Income: The top two income tax brackets would return to their
     * 1990's levels of 36% and 39.6%. All other tax brackets would remain as
     * they are today." According to the tax code, the top two brackets
     * correspond to $164,551 and above in 2008, meaning these Obama effects
     * will be felt by people earning over $164,000. These people roughly
     * correspond with the top 5% of the population, since according to the US
     * Census Bureau in 2006 (see table at
     * http://en.wikipedia.org/wiki/Income_in_the_United_States) the top 5%
     * earns $167,000 and up. (Since these earnings have obviously increased in
     * 2008, this goes over $164,551 a bit, but this is fine, since not all
     * income is taxed, which corrects for the error.) Thus, the average tax for
     * the top 5% of people was made to return to 1990's levels to reflect
     * Obama's income tax hike. The average 1990's level for the top 5% was
     * around 24% according to Source 1, so that is all that was changed in the
     * tax array to account for Obama's plans, since he stated other tax
     * brackets would remain the same. Notice that this is only a good
     * approximation since not all income tax for people who earn in the top
     * bracket is calculated using the top bracket: other brackets come into
     * play, too. These other brackets were slightly reduced during the Bush
     * administration, so one could argue that the effective tax rate for the
     * top 5% would actually be slightly lower than it was in the 1990's if
     * Obama's tax plans were implemented. So maybe tax[0] should be a little
     * over 24, but in that case only a little.
     */

    BUSH(new double[] {
            20.68, 12.60, 9.36, 7.01, 3.01 }, "Bush's tax plan"),
    OBAMA(new double[] {
            24.00, 12.60, 9.36, 7.01, 3.01 }, "Obama's tax plan"),
    CHANGE(new double[] {
            27.00, 11.60, 8.36, 6.01, 2.01 }, "our tax plan");

    double[] tax;
    String name;

    TaxPlan(double[] tax, String name)
    {
        this.tax = tax;
        this.name = name;
    }

    public String toString()
    {
        return name;
    }
}
```

```
public enum WarPlan
{
    //according to the table S-3
(http://www.whitehouse.gov/omb/budget/fy2009/summarytables.html),
    //  Department of Defense spending is $479.5 billion
    //the war in Afghanistan began in 2001, and the
    //  warn in Iraq began in 2003
    //an Iraq withdrawal would return DoD funding to
    //  2002 levels, and an Iraq/Afghanistan withdrawal
    //  would return DoD funding to 2000 levels
    //  (plus regular inflation per year)
    //historical funding levels were found in
    //  table 3.2 at http://www.whitehouse.gov/omb/budget/fy2009/pdf/hist.pdf
    STATUS_QUO
    {
        public double getSpending(DebtSimulation sim)
        {
            //account only for inflation
            return 479.5 * Math.pow(sim.INFLATION, sim.time - 2008);
        }
        public String toString()
        {
            return "Maintain the status quo in Iraq";
        }
    },
    OBAMA_WITHDRAWAL
    {
        public double getSpending(DebtSimulation sim)
        {
            //will withdraw troops from Iraq mid-2010,
            //  will keep troops in Afghanistan
            //so, by 2011, DoD spending will return to
            //  2002 levels (plus inflation)
            double DoD2011 = 348.5 * Math.pow(sim.INFLATION, 2011 - 2002);
            if (sim.intTime >= 2011) //account for inflation
                return DoD2011 * Math.pow(sim.INFLATION, sim.time - 2011);
            //otherwise, draw the line between (2008,479.5) and
            //  (2011,DoD2011) and plot the point along that line
            return 479.5 + (sim.time - 2008) * (DoD2011 - 479.5) / (2011 - 2008);
        }
        public String toString()
        {
            return "Withdraw from Iraq";
        }
    },
    SURGE
    {
        public double getSpending(DebtSimulation sim)
        {
            //linear regression on national defense spending
            //  from 2001-2007 (table 3.2,
http://www.whitehouse.gov/omb/budget/fy2009/pdf/hist.pdf)
            //  gives (where x=years since 2000):
            //  42.16753571x+271.8441429 (R^2=.9920709579)
            return 42.16753571 * (sim.time - 2000) + 271.8441429;
        }
        public String toString()
        {
            return "Continue the troop surge";
        }
    },
    AFGHANISTAN_WITHDRAWAL
    {
```

```
        public double getSpending(DebtSimulation sim)
        {
            //if we were to withdraw all troops from Iraq
            //  and Afghanistan by mid-2010, then by 2011,
            //  DoD spending will return to 2000 levels
            //  (plus inflation)
            double DoD2011 = 294.4 * Math.pow(sim.INFLATION, 2011 - 2000);
            if (sim.intTime >= 2011) //account for inflation
                return DoD2011 * Math.pow(sim.INFLATION, sim.time - 2011);
            //otherwise, draw the line between (2008,479.5) and
            //  (2011,DoD2011) and plot the point along that line
            return 479.5 + (sim.time - 2008) * (DoD2011 - 479.5) / (2011 - 2008);
        }
        public String toString()
        {
            return "Withdraw from Iraq and Afghanistan";
        }
    };

    public abstract double getSpending(DebtSimulation sim);
    public abstract String toString();
}
```

```
public class Population
{
    //people[yr][0] is how many people were born in "yr"
    //  (so they are less than one year old)
    //people[yr][age] is how many people are "age" years old
    //  (for 1<=i<=84) in year "yr"
    //people[yr][85] is how many people are 85 years old
    //  or older in year "yr"
    //where year 0 refers to 2006
    int people[][];

    public Population()
    {
        people = new int[12][86];
        initializeData();
        for (int i = 1; i < 12; i++)
            calculateData(i);
    }

    public int getNumberOfPeople(int year, int age)
    {
        if (age < 0 || age > 85)
            throw new IllegalArgumentException("0 <= age <= 85");
        if (year < 2006 || year > 2017)
            throw new IllegalArgumentException("2006 <= year <= 2017");
        return people[year - 2006][age];
    }

    public int getNumberOfPeople(int year, int minAge, int maxAge)
    {
        int numberOfPeople = 0;
        for (int i = minAge; i <= maxAge; i++)
            numberOfPeople += getNumberOfPeople(year, i);
        return numberOfPeople;
    }

    public int getTotalNumberOfPeople(int year)
    {
        return getNumberOfPeople(year, 0, 85);
    }

    private void initializeData()
    {
        //data for people age 5...85 is for 2006 and
        //  from http://www.census.gov/popest/national/asrh/NC-EST2006/NC-EST2006-01.xls
        //  and uniformity was assumed within each age group
        //for for infants age 0...4, death rate data
        //  (source given below) was used to back-calculate
        //  population sizes from death rates and death
        //  numbers for infants age 0 and infants age 1...4
        //  (with uniformity assumed in the latter group)
        //censusData[i] describes how many people there
        //  are aged censusRanges[i][0]...censusRanges[i][1]
        int censusData[] = {
                4106859, 16176871, 19709887, 20627397, 21324186, 21111240,
                20709480, 19706499, 21185785, 22481165, 22797569, 20480605,
                18224445, 13362238, 10375554, 8541290, 7381027, 5665664,
                5296817 };
        int censusRanges[][] = {
                {
                        0, 0 }, {
                        1, 4 }, {
                        5, 9 }, {
```

```
                        10, 14 }, {
                        15, 19 }, {
                        20, 24 }, {
                        25, 29 }, {
                        30, 34 }, {
                        35, 39 }, {
                        40, 44 }, {
                        45, 49 }, {
                        50, 54 }, {
                        55, 59 }, {
                        60, 64 }, {
                        65, 69 }, {
                        70, 74 }, {
                        75, 79 }, {
                        80, 84 }, {
                        85, 85 } };
        for (int i = 0; i < censusData.length; i++)
            for (int j = censusRanges[i][0]; j <= censusRanges[i][1]; j++)
                people[0][j] = censusData[i] / (censusRanges[i][1] - censusRanges[i][0] +
 1);
    }

    //calculate the population for year "yr"
    //  (assuming population for year "yr"-1
    //  has already been calculated)
    private void calculateData(int yr)
    {
        //first, everyone ages by one year
        people[yr][85] = people[yr - 1][85] + people[yr - 1][84];
        for (int i = 84; i >= 1; i--)
            people[yr][i] = people[yr - 1][i - 1];
        people[yr][0] = 0;

        //next, babies are born according to the
        //  birth rate (assuming women are 50%
        //  of each age group)
        //data for birth rates from
ftp://ftp.cdc.gov/pub/Health_Statistics/NCHS/Dataset_Documentation/DVS/natality/UserGuide2
005.pdf (table 3)
        //birthRates[i] describes the birth rate per 1,000 women
        //  aged birthRanges[i][0]...birthRanges[i][1]
        double birthRates[] = {
                0.7, 21.4, 69.9, 102.2, 115.5, 95.8, 46.3, 9.1, 0.6 };
        int birthRanges[][] = {
                {
                        10, 14 }, {
                        15, 17 }, {
                        18, 19 }, {
                        20, 24 }, {
                        25, 29 }, {
                        30, 34 }, {
                        35, 39 }, {
                        40, 44 }, {
                        45, 49 } };
        for (int i = 0; i < birthRates.length; i++)
            for (int j = birthRanges[i][0]; j <= birthRanges[i][1]; j++)
                people[yr][0] += (int) (birthRates[i] * people[yr][j] / 2 / 1000);

        //finally, death rates are used to calculate
        //  the deaths in each age group
        //data for death rates from http://www.disastercenter.com/cdc/Death%20rates%
202005.html
        //deathRates[i] describes the death rate per 100,000 people
```

```
        //  aged deathRanges[i][0]...deathRanges[i][1]
        double deathRates[] = {
                692.5, 29.4, 14.5, 18.1, 65.1, 97.6, 97.5, 111.4, 149.6, 233.4,
                353.1, 520.8, 734.6, 1136.9, 1700.0, 2657.6, 4154.0, 6712.9,
                13798.6 };
        int deathRanges[][] = {
                {
                        0, 0 }, {
                        1, 4 }, {
                        5, 9 }, {
                        10, 14 }, {
                        15, 19 }, {
                        20, 24 }, {
                        25, 29 }, {
                        30, 34 }, {
                        35, 39 }, {
                        40, 44 }, {
                        45, 49 }, {
                        50, 54 }, {
                        55, 59 }, {
                        60, 64 }, {
                        65, 69 }, {
                        70, 74 }, {
                        75, 79 }, {
                        80, 84 }, {
                        85, 85 } };
        for (int i = 0; i < deathRates.length; i++)
            for (int j = deathRanges[i][0]; j <= deathRanges[i][1]; j++)
                people[yr][j] -= (int) (deathRates[i] * people[yr][j] / 100000);
    }
}
```

```java
import java.io.*;

public class DataLoader
{
    private DataLoader()
    { // can't instantiate this class.
    }

    public static double[][] readFile(String filename) throws IOException
    {
        StreamTokenizer in = new StreamTokenizer(new BufferedReader(new
 FileReader(filename)));
        in.nextToken();
        int rows = (int) in.nval;
        in.nextToken();
        int cols = (int) in.nval;

        double[][] data = new double[rows][cols];
        for (int i = 0; i < rows; i++)
            for (int j = 0; j < cols; j++)
            {
                in.nextToken();
                data[i][j] = in.nval;
            }

        return data;
    }
}
```

```
public abstract class State
{
}
```

```java
public abstract class TimeSimulation<T extends State>
{
    public double time;
    public int intTime;
    public final double dt;
    public final double starttime;
    public final double endtime;

    public T current;
    public T next;

    public TimeSimulation(double starttime, double endtime, double dt)
    {
        this.starttime = starttime;
        this.endtime = endtime;
        this.dt = dt;
    }

    public void run()
    {
        initialize();
        for (time = starttime, intTime = (int) Math.round(time); time <= endtime; time += dt, intTime = (int) Math.round(time))
        {
            update();
            T temp = current;
            current = next;
            next = temp;
        }
        finish();
    }

    /**
     * Initialize is what occurs at the beginning of the program.
     */
    public abstract void initialize();

    /**
     * Update is the action that occurs during each time step.
     */
    public abstract void update();

    /**
     * Finish is whatever happens after the simulation is complete.
     */
    public abstract void finish();

    /**
     * Show statistics from the most recent run.
     */
    public abstract void showStats();
}
```

Team #2094
November 2008

Dear President-elect Barack Obama,

The growing national debt presents serious risks to the American economy, our quality of life and ultimately our national security. It is absolutely vital that your policies reflect a thorough understanding of this growing problem.

Currently, the government owes about seventy percent of the U.S. GDP to others. This is a problem for three reasons. First, a large portion of the US budget must be devoted to paying interest on money it has borrowed. A growing debt means eventually the US government would need to raise taxes significantly to pay off interest, crippling the economy. Second, increased government borrowing would increase interest rates, making it harder for private individuals and corporations to gain capital. This would stifle economic growth. Third, the government owes much of its debt to foreign nations. According to Milton Friedman, Nobel-prize-winning economist, a U.S. dependence on foreign nations for funding could hinder our nation's capability to act on its own.

To better understand how national debt could increase, we made a computer model of the US budget to crunch the numbers and see how different policies would affect the national debt. The model considered expenditure policies (including defense spending), tax policies and the effect of an aging population on social security. Continuing current wars and Bush tax policies on our model, we find that national debt balloons at 2017. Increased social security spending contributes to this. With such a debt, interest payments increase greatly from their current level of around ten percent of the budget revenue, according to our model. Clearly, debt should be controlled to lesser levels to keep interest payments down and mitigate effects of massive debt discussed earlier in the letter. Therefore, we favor the new income tax plans you have proposed, which according to our model would cause debt to go down to 37.3% percent of GDP, a substantial improvement. However, excessive tax increases to balance the budget should be weighed against possible adverse affects to the economy.

The best change to lower national debt is lowering defense spending by following your ideas to withdraw from Iraq. The less we fight, the more money we have and the less debt there will be. In fact, our model predicts that by withdrawing from Iraq (and keeping your new tax policies), 2017 debt becomes far more manageable then it would be if defense spending continues at its current rate.

We realize that you seek to implement social programs that would increase expenditures. These possible increases in spending should occur only if our current wars can reasonably quickly be brought to an end. Otherwise, debt would become extremely unmanageable. Before making spending changes, take care to consider how the would affect the national debt: our flexible model is an ideal tool.

We hope you will carry out your campaign promises of trying to get a little more tax money out of the wealthy and ending wars. If you carry out those promises, we, as the designers of a national debt model, believe that they will enable the government to remain financially prudent while it carries out some of the other promises you have made that increase spending.

Sincerely,

Team #2094