# Accumulation and Withdrawal Program Specification*

Jeffrey D. Oldham

2000 Apr 18

We sketch the withdrawal and accumulation program specification, outlining its input, output, and computation.

## 1 Revisions

2000Apr27: Added age to C++ program input specification. Added specification of C++ program's output.

2000Apr25: Added financial target inflation boolean to the input.

2000Apr18: Changed the monthly modification specification from proportional splitting to a 50%/50% split between capital appreciation and income return. Rationale: Proportional splitting is hard to specify if one return rate is positive and one is negative. Suggestions for a more realistic arrangement are solicited.

2000Apr18: Added input specification for the C++ program to the end of Section 2.

## 2 Program Outline

The program consists of three separate pieces:

- a WWW interface taking input from the user and displaying the results,

- a CGI program checking the input for correct values, invoking the underlying program, and sending its results as a WWW page, and
- the withdrawal and accumulation program using historical data to determine the probability of reaching one's financial goal.

Throughout the specification, we note possible features for future versions of the program. We need not program these for this version.

# 3 Input

We specify the input from the user's point of view, i.e., what the WWW interface provides. The user needs to specify

- the monetary amounts,
- time periods,
- asset allocations, and
- tax information.

## 3.1 Monetary Amounts

The user should specify the monetary amounts:

**initial amount:** the initial amount of money to invest. For the withdrawal problem, this is the monetary amount of investments. A default value of $0 will ease input for the accumulation problem.

**increment amount:** the amount of money to invest every time period. A negative amount indicates a periodic withdrawal. A default value of $100 might be nice.

**increment inflation:** a boolean indicating whether the increment amount should be inflated by the Consumer Price Index (CPI). A value of true indicates the increment amount should be increased every month by the monthly CPI. A value of false indicates the amount should remain unchanged during the entire simulation. A default value of true will ease input for the withdrawal problem.

**financial target:** the desired ending amount of money. For the withdrawal problem, this is probably $0, i.e., the ending amount of money is at least $0. A default value of $0 will ease input for the withdrawal problem.

**financial target inflation:** a boolean indicating whether the desired ending amount of money should be inflated by the CPI. For the withdrawal problem, the financial target is probably $0 so the boolean's value is moot. For the accumulation problem, specifying a true value will indicate the financial target is specified in current dollars and a false value will indicate the target is in future dollars. A default value of true will ease input.

All monetary amounts should be rounded to the nearest dollar. Only the increment amount may be negative.

## 3.2 Time Periods

The user should specify the time periods:

**time frame:** the length of time to invest the money. A default value of twenty-five years may ease input for the withdrawal problem.

**increment time period:** the length of time between periodic investments. At the end of each increment time period, the increment amount (inflated if appropriate) is added to the investment. The time frame need not be a multiple of the increment time period. A default value of twelve months may ease input.

Time periods must have positive length. Since we will use monthly historical data, the smallest unit for time periods should be one month. The WWW page designers should produce the clearest interface permitting time periods up to the nearest month.

In an advanced version, we may permit specifying the range of historical data to use. For example, users might specify simulating using post-WWII data only.

The initial money is assumed to be invested at the beginning of the time frame. The first increment occurs at the end of the first time period. All investments are sold at the end of the time frame. For example, consider a time frame of twenty-three years starting in 1926 with an increment time period of five years. The initial investment is made 1926 Jan 01. Incremental investments occur on the last day of 1930, 1935, 1940, and 1945. The entire investment is sold on the last day of 1948.

## 3.3 Asset Allocations

The user should specify the desired asset allocation. Every rebalancing time period, assets are sold or purchased to move the investment back to the desired allocation.

Some users may wish to specify a fixed asset allocation, choosing among the following assets:

**large company stocks:** stocks of the largest U.S.-based companies. For recent years, this is just the S&P 500.

**long-term government bonds:** bonds usually with a twenty-year maturity invested in the U.S. federal government.

**intermediate-term government bonds:** bonds with at least a five-year maturity invested in the U.S. federal government.

**U.S. Treasury bills:** in financial jargon, "cash." This is invested in the shortest period risk-free U.S. government debt with at least one month in maturity. As cash, the bills provide no capital appreciation.

The asset allocations should be specified as integral percentages totalling to 100. Default values of $(60, 0, 30, 10)$ might ease user input.

Alternatively, the user could specify one of three age-dependent formulae:

| strategy | stock allocation | LT government bond allocation |
|---|---|---|
| conservative | $\min(125 - \text{age}, 100)$ | remainder |
| moderate | $\min(135 - \text{age}, 100)$ | remainder |
| aggressive | $\min(145 - \text{age}, 100)$ | remainder |

Thus, the user should also supply a starting age.

The historical data for small company stocks and long-term corporate bonds does not provide separate capital appreciation and income returns so we omit these asset classes from our simulation. An advanced version might include small company stock data, assuming no income return. An advanced version computing for tax-free accumulation or withdrawal might also include long-term corporate bonds.

## 3.4 Tax Information

The program will simulate investment growth in either a tax-free or a taxable account, as specified by the user. If the investment is held in a taxable account, the user should specify:

**long-term capital gains tax rate:** the tax rate for long-term capital gains. For simplicity, the long-term capital gains tax rate is applied to any asset held (strictly) longer than one year, while the short-term rate is applied for any asset held twelve months or less. For simplicity, the short-term rate equals the ordinary income tax rate. The rate must be in the range $[0, 100]$. A default value of 20, i.e., a rate of 20%, might ease user input.

**ordinary income tax rate:** the tax rate for short-term capital gains, dividends, and ordinary income. The rate must be in the range $[0, 100]$. A default value of 28, i.e., a rate of 28% might ease user input.

**holding time period for assets:** the length of time to hold an asset before it is sold. For example, the user could specify all stocks and bonds are held for five years and then sold. Alternatively, we could have specified an average annual turnover rate, i.e., what fraction of the portfolio is sold every year.

The program assumes:

- All investments are sold at the end of the time frame with the appropriate taxes paid.
- All taxes are paid when an asset is sold even if this is during the middle of the calendar year.
- Selling assets at a loss does not incur any taxes, but the loss does not reduce the taxes on income or future year's taxes.

A future version might use historical tax information to 1) provide more accurate calculations and 2) show whether market returns and tax rates are correlated. It may also incorporate the actual historical capital gains tax rules, which have varied through time.

## 3.5 Sensitivity Analysis

The WWW page need not ask the user for sensitivity analysis input. A future version may require input.

The underlying computation should support modifying the historical monthly return rates by a given fixed percentage. A positive number indicates exceeding historical return rates, while a (more likely) negative number indicates not attaining the historical rates. For assets with return rates specified separately as capital appreciation and income returns, the percentage should be split evenly.

As an aside, the annual modification amount $M$ technically depends on the monthly returns according to the formula:

$$M \approx m \left( \frac{1}{1 + r_{\text{jan}}} + \ldots + \frac{1}{1 + r_{\text{dec}}} \right),$$

where $m$ is the monthly modification amount. For our purposes, we will approximate this formula as $M = 12m$.

### 3.6 C++ Program Input Specification

The C++ program's input should be read from the command-line arguments. See Table 1. Note that either the choice of the age-varying asset allocation formula and the age xor the fixed asset allocations are specified in the input. Also, the tax rates are omitted if taxes are not being computed. Collin and I decided to omit the database file names. The C++ program and not the CGI program will probably know their location.

Craig recommends that all input processing be performed inside some function to modularize the code.

## 4 Output

The withdrawal and accumulation program produces the probability of meeting or exceeding the financial goal and also produces the average and standard deviation of the time frames' investment amounts. The probability, specified as the ratio of time frames meeting or exceeding the goal to the number of time frames, should be displayed as an integral percentage with maximum value 100%. To avoid misleading the user, the average and standard deviation should perhaps be rounded to three significant digits.

Probabilities should be displayed assuming

- the user's investment actually attains the historical rate data,
- the user's investment has a monthly modification amount of -0.03%, and
- the user's investment has a monthly modification amount of -0.10%.

A future version might display histograms of the ending amounts.

| | | |
|---|---|---|
| initial investment amount | unsigned long | |
| investment increment amount | long (positive, zero, or negative) | |
| increment by inflation? | 0 = no monthly inflation increment, 1 = inflation increment | |
| desired investment target | unsigned long | |
| financial target inflation | 0 = no monthly inflation, 1 = inflate target | |
| time frame | months as positive long | |
| increment time frame | months as positive long | |
| age-varying asset allocation | 0 = fixed, 1 = age-varying | |
| age-varying formula | 1 = conservative, 2 = moderate, 3 = aggressive | if varying |
| age | unsigned long (age in years) | if varying |
| large-company allocation | unsigned long in range $[0, 100]$ | if fixed |
| long-term govt bond allocation | unsigned long in range $[0, 100]$ | if fixed |
| intermediate-term govt bond allocation | unsigned long in range $[0, 100]$ | if fixed |
| Treasury bill allocation | unsigned long in range $[0, 100]$ | if fixed |
| taxes? | 0 = no taxes, 1 = taxed | |
| long-term capital gains tax rate | nonnegative integer in range $[0, 100]$ | if taxed |
| ordinary income tax rate | nonnegative integer in range $[0, 100]$ | if taxed |
| asset-holding time | months as positive long | if taxed |
| monthly rate modification | double (as a percentage) | |
| file of stock cap. apprec. returns | string | |
| file of stock income returns | string | |
| file of LT govt. bonds capital appreciation returns | string | |
| file of LT govt. bonds income returns | string | |
| file of IT govt. bonds capital appreciation returns | string | |
| file of IT govt. bonds income returns | string | |
| file of U.S. Treasury bill returns | string | |
| file of inflation data | string | |

Table 1: C++ Program Input Specification

### 4.1 Output from the Computation Program

For one set of input, the C++ program will produce the following output all on one line.

- an integer indicating success, i.e., 0, or failure, i.e., a nonnegative integer not 0,
- the probability of success as an integer in the range $[0, 100]$,
- the average ending investment value as a nonnegative integer with three significant digits,
- the standard deviation of the ending investment value as a nonnegative integer with three significant digits.

## 5 Computation

First, we will describe a computation without capital gains taxes. Then, we will revise the computation to incorporate capital gains taxes.

### 5.1 Dealing with the Historical Data

The program should compute the probability of success and the average and standard deviation over all time frames. So that the historical return data need not be repeatedly read from disk, it could be read into STL vectors. Secondly, this data might be packaged in a class with a function returning the historical rate for a specified asset, year, month, and income or capital appreciation. Using such a class will hopefully avoid indexing problems.

A statistics class accumulating the probability of success, the average, and the standard deviation could ease collecting this data. Its constructor could take the desired target. Every time an additional piece of data is computed, a class member function could be invoked. Its destructor could print the statistics.

In the text below, we will concentrate on determining the final investment value for a particular time frame. We will assume the program will loop through all possible time frames.

## 5.2 Tax-free Accumulation and Withdrawal

To compute the final tax-free accumulation or withdrawal amount, useful values include

**total investment value:** the total value of all investments at any particular time

**current asset allocation:** a mapping from asset type to the percentage invested in the particular asset. For example, stocks could form 0.60 fraction of the total investment. It might be useful to make a C++ asset allocation class. This class could include the person's age if appropriate.

**increment amount:** the amount to increment the investment. This amount is inflated if specified by the user.

**months until incrementing:** the number of months until the investment should be incremented.

**months until finish:** the number of months until the time frame ends.

The steps to perform at the time frame's beginning include:

1. Set the initial asset allocation.
2. Set the total investment value and the increment value to the appropriate values.
3. Set the months until incrementing and the months until finishing to the appropriate values.

The steps to perform at the end of each month and beginning of the next month include:

1. Increase the investment in each asset by the asset's return that month.

   $$\text{monthly rate} = \text{capital appreciation} + \text{income rate} + \text{monthly modification}$$

2. Inflate the increment amount if appropriate.
3. If the investment should be incremented, increase its value and reset the number of months until the next increment.
4. Decrement the number of months until the time frame ends.

5. Update the asset allocation according to the formula if using an age-dependent formula.

When the simulation of this time frame ends, record the total investment's worth.

## 5.3 Taxed Accumulation and Withdrawal

(This presentation is different that that discussed in class Thursday, 13 April. I conjecture this approach is more precise than the approximation discussed in class.)

Taxes on income could easily be incorporated by modifying the monthly return rate to pay taxes monthly. Taxes on appreciated assets are more complicated because the capital gains tax due is the product of the increase in the asset's value and the capital gains tax rate. Thus, we will need to remove the basis of all purchased assets subject to capital appreciation.

### 5.3.1 Values and Data Structures

Useful values and data structures include

**current asset allocation:** a mapping from asset type to the percentage invested in the particular asset. For example, stocks could form 0.60 fraction of the total investment. It might be useful to make a C++ asset allocation class. This class could include the person's age if appropriate.

**increment amount:** the amount to increment the investment. This amount is inflated if specified by the user.

**months until incrementing:** the number of months until the investment should be incremented.

**months until finish:** the number of months until the time frame ends.

For each asset subject to capital gains, we should record the month of each purchase, its purchase price, and its current value. When the difference between the current month and a purchase's date equals the asset holding time period, that purchase should be sold. Each month's purchases is added to the other end of the queue. Recording the total value of each asset simplifies some calculations. Cash, i.e., U.S. Treasury bills, are not subject to capital appreciation.

The steps to perform at the time frame's beginning include:

1. Set the initial asset allocation.
2. For each asset, purchase the appropriate amount of assets.
3. Set the increment value to the appropriate value.
4. Set the months until incrementing and the months until finishing to the appropriate values.

### 5.3.2 Each Month's Computation

The steps to perform at the end of each month and the beginning of the next month mimic the actions a human investor would perform.

1. Income from each asset is computed using the assets' income return rate and the user's ordinary income tax rate. For example, the income from the large company stock holding is the product of

   - the total large company stock value,
   - the stock income return rate plus a proportion of the monthly modification amount, and
   - one minus the ordinary income tax rate.

   Since this income is received as cash, this income is added to the U.S. Treasury bill asset, but be sure to compute the income on the U.S. Treasury bill asset first.

2. Asset values are increased by the capital appreciation rate. That is, the current values of all assets are multiplied by the asset's capital appreciation return rate plus a proportion of the monthly modification amount.
3. Asset purchases held for the entire asset holding period are sold and capital gains taxes are paid. The amount received equals the asset's current value. The capital gains tax equals the product of the capital gains tax rate and the difference between the asset's current value and its purchase price. If the difference is negative, the tax is zero. Proceeds are paid in cash so they are added to the U.S. Treasury bill asset.
4. Inflate the increment amount if appropriate.

5. If the investment should be incremented, increase the Treasury bill asset and reset the number of months until the next increment.

6. Decrement the number of months until the time frame ends.

7. Update the asset allocation according to the formula if using an age-dependent formula.

8. Redistribute the assets to meet the asset allocation. The presence of capital gains taxes complicates this calculation.

### 5.3.3 Reallocating Assets in the Presence of Capital Gains Taxes

Overallocated assets, i.e., assets exceeding their allocation, should be sold to purchase underallocated assets. Purchasing assets incurs no penalties, while selling assets requires paying capital gains taxes. The amount of capital gains taxes to pay depends on each purchase's initial price.

A series of formula manipulations yields a set of linear equations to solve. Underallocated assets are omitted from the linear equations. Assume assets types 1, 2, and 3 are overallocated. Let $A_i$ be the desired asset allocation fraction, e.g., 0.60, for each type $i$. Let $a_i$ represent the total dollar amount invested in the asset. Let $t$ represent the total dollar amount in all assets. Let $g_i$ represent the marginal capital gains tax rate for the oldest purchase in asset type $i$. For example, suppose the initial price of the oldest purchase may have been \$100, its current value is \$150, and the capital gains tax rate is 20%.[1] The marginal capital gains tax rate is the ratio of the tax paid if all of the purchase is sold to the purchase's value, i.e., the ratio of $0.20 * \max(0, 150 - 100)$ and $150$. Let $f_i$ represent the fraction of the asset to sell.

Solve this set of linear equations for $f_i$.

$$\begin{bmatrix} (A_1-1)(1-g_1)a_1 & A_1(1-g_2)a_2 & A_1(1-g_3)a_3 \\ A_2(1-g_1)a_1 & (1-A_2)(1-g_2)a_2 & A_2(1-g_3)a_3 \\ A_3(1-g_1)a_1 & A_3(1-g_2)a_2 & (1-A_3)(1-g_3)a_3 \end{bmatrix} \begin{bmatrix} f_1 \\ f_2 \\ f_3 \end{bmatrix} = \begin{bmatrix} A_1t - a_1 \\ A_2t - a_2 \\ A_3t - a_3 \end{bmatrix}$$

Among all the assets with $f_i a_i$ greater than the oldest purchase's current value, sell the asset with the smallest marginal capital gains tax rate so minimize the tax paid. If this occurs, form a new set of linear equations and repeat the calculation.

[1] Use the ordinary income tax rate if the asset was held twelve months or less.

Finally, all the $f_i a_i$ products are less than or equal to asset's oldest purchases' current values, sell $f_i a_i$ from each asset, adding the money to the U.S. Treasury bill asset. Then purchase underallocated assets using cash from the U.S. Treasury bill asset. The previously overallocated asset amounts should remain unchanged.