GENG 106 Computer Programming
Spring 2010
Cooperative Project

<u>Due Date: 23rd May 2010 – Before 12:00 Noon</u>

Problem: Asset Depreciation

The chief accountant at Qatar Ready-mix needs to calculate the depreciation value for the fixed and mobile assets in the company (batching plants, ice plants, cement pumps, …etc.).

Depreciation is a decrease in the value over time of some asset due to wear and tear, decay, declining price, and so on. For example, suppose that the company purchases a new computer system for 200,000 QR that will serve the company's needs for 5 years. After that time, called the *useful life* of the computer, it can be sold at an estimated price of 50,000 QR, which is the computer's *salvage* value. Thus, the value of the computing equipment will have depreciated 150,000 QR over the 5-year period.

The chief accountant wants to adopt two methods to calculate depreciation: *straight-line* method for the fixed assets and *sum-of-years-digits* method for the mobile assets.

Straight-line method:

The amount to be depreciated is divided <u>evenly</u> over a specified number of years that resemble the useful life of the asset. For example, a straight-line depreciation of 200,000 QR over a 5-year period gives an annual depreciation of 200,000/5 = 30,000 QR.

Year	**Depreciation**
1	30,000 QR
2	30,000 QR
3	30,000 QR
4	30,000 QR
5	30,000 QR

Sum-of-years-digits method:

The amount to be depreciated is divided proportionally over a specified number of years in the way demonstrated in the following example:
Consider again depreciating 150,000 QR over a 5-year period. We first calculate the "sum of the years digits", 1 + 2 + 3 + 4 + 5 = 15. In the first year, 5/15 of 150,000 QR

(50,000 QR) is depreciated; in the second year, 4/15 of 150,000 QR (40,000 QR) is depreciated; and so on, giving the following depreciation table:

Year	Depreciation
1	50,000 QR
2	40,000 QR
3	30,000 QR
4	20,000 QR
5	10,000 QR

What is needed?

To help with computing the depreciation, you need to develop a menu driven program that will allow the chief accountant to choose one of the following tasks:

1. **Create an assets file.**
2. **Calculate depreciation, create a depreciation schedule file and display depreciation tables.**
3. **Append a new asset to the assets file.**

1. **<u>Create an assets file</u>**

   When the user chooses 1, the program should perform the following tasks:

   a. Prompt the user to enter an assets file name and create that file.

   b. Allow the user to enter the below data (without the title row) into the assets file.

Asset Code	Asset name	Type	Quantity	Useful life (years)	Asset value (QR)	Salvage value (QR)
100	Cement pump	Mobile	2	5	2,500,000	375,000
200	Batching plant	Fixed	2	7	2,000,000	300,000
300	Ice plant	Fixed	1	8	1,500,000	225,000
400	Power generator	Fixed	2	9	750,000	112,500
500	Transit mixer	Mobile	4	6	500,000	75,000
600	Chiller	Fixed	1	7	500,000	75,000
700	Cement bulker	Mobile	2	5	425,000	63,750
800	Shovel	Mobile	2	6	350,000	52,500
900	Forklift	Mobile	2	7	200,000	30,000

2. **Calculate depreciation, create the depreciation schedule file and display depreciation tables.**

   When the user chooses 2, the program should perform the following tasks:

   a. Prompt the user to enter the name of the asset depreciation schedule file and create it.
   b. Prompt the user to enter the name of the assets file (created in choice 1).
   c. Read the assets file. For each asset in the assets file, the program should calculate the depreciation value across the useful life years of that asset as described in the problem above. Your program should have two functions to calculate depreciation (refer to the computer system depreciation example in the problem description) :
      - `straightLine()` function should be called to calculate the depreciation of fixed assets.
      - `sumOfYearsDigits()` function should be called to calculate the depreciation of mobile assets.
   d. You should store the depreciation tables to the depreciation schedule file (created in step a).
   e. Display the depreciation tables on the screen.
   f. Calculate and display a summary depreciation table that will sum up the depreciation values of all assets across their useful life years.

      <u>**BONUS**</u>: there will be a bonus for those who use an array in calculating the sum of depreciation values across the years.

3. <u>**Append new assets to the assets file.**</u>

   When the user chooses 3, the program should perform the following tasks:

   a. Prompt the user to enter the name of the assets file.
   b. Open the assets file in append mode so that not to destroy the file.
   c. Prompt the user to enter the asset code and the rest of the data (as in choice 1) and write them to the assets file. Keep looping until a (-1) Sentinel is entered for the asset code.

Guidelines and Timeline:

- There should be 2 members in each team. You should send me the names of the team members, and a "nickname" for the team by **27th April 2010**
- Team members should meet regularly for discussions and workload distribution - at least once a week starting now until the project is finished.
- Submission of project is strictly due on **23rd May 2010.** **You should follow the submission instructions at the end of this document**.
- **The role of each team member must be clearly identified using the attached Cooperative Project Evaluation Form (CooperativeProjectEvaluationForm.doc). Each team member should submit a filled copy of this form signed by all members of that team, where designated.**

  Note: Your project will not be graded if this form is not submitted.

Cooperative Project Evaluation Criteria (Rubric):

The grade for the project is 10 points distributed as follows:

1. Correctness and Efficiency (45%)
   - Use of functions.
   - Use of files and loops
   - Error handling when opening files and when reading from files.
   - Validation of user input when entering menu choices and looping.
   - Correctness of application execution and results
2. Team Cooperation and Coordination (15%)
   - Evident and clear role of each member (use the attached form)
3. Documentation & user friendliness (15%)
   - Name of the program and what the program should do.
   - Who wrote the program?
   - Problem specification (i.e. the `input` and `output` clauses)
   - Function specification (i.e. the `receives` and `returns` clauses)
   - Meaningful identifiers
   - In program comments where necessary
   - White space and proper indentation
   - Informative input prompts
   - Labeled output
4. Discussion of project (25%)
5. Lateness penalty (-10% per day)
6. **Copying and/or plagiarism (-100%)**

Submission Instructions:

- You should turn in: 1) your source code (as .cpp files **not** .txt); 2) a sample run of your application; and 3) a filled and **signed** "Cooperative Project Evaluation" form, **both electronically via Blackboard AND** in **PRINT** form.
- For the electronic submission, you should include the three items above in a zipped folder that should adopt the following naming convention:

  GENG106-Project-[*day & letter month of your submittal date*]-[*nickname of the team*]
  e.g. if students X and Y have teamed; and they have given the nickname "Smarties" to their team; and they have submitted their project on 1st January, then the folder name should be:
  GENG106-Project -1Jan-Smarties