

## Semester project

Final version due ----; project is submitted in preliminary versions during the semester

You are to design, code in Java, and test a simple menu-driven contact-management application, as specified below.

### Specification

1. Your application will allow the user to manage information about a collection of contacts (*records*, in database terminology), stored in a disk file.

A contact has the following attributes:

- Name;
- Phone number;
- Email address.

2. Your application will display a menu of user options. The initial version will manage just one contact and will have the following choices:

- Input contact;
- Display contact;
- Update contact;
- Save contact to disk;
- Retrieve contact from disk.

Your application will perform, or allow the user to perform, the operation corresponding to the chosen menu item. Display an error message if user requests output of data before inputting or retrieving data.

3. All input data should be *validated* upon input. For example, negative input values should result in error messages. A special challenge is to implement the saving of multiple contacts in your file and to enable searching and display of this file.

4. The *Display* operation will display a contact in a tabular format, as below, including the total value of all the instances of the item in stock:

Name	phone	email
----	-----	-----
Grace Hopper	212-200-5432	grace@hopper.com

5. After our discussion on arrays, extend your application to store your contact list as an array of objects and to maintain a disk file of multiple contacts. In that version, the application will have the following menu choices:

- New item (append to file);
- Search by ID (if found, display the item and display a menu to allow update or delete the item);
- Display all items in collection, including summary information;
- Generate a report file in same format as used by the display option.

If you wish to work with arrays as well (not part of this course), you may implement a collection of records using an array.

### Implementation

1. Begin by writing *several clear paragraphs* that summarize:

- what the program does (specification), i.e., how the program interacts with the user;
- how it works internally (design).

This section should include *flowcharts*, *module hierarchy diagrams*, and *UML class diagrams*.

2. Initially your project will consist of separate programs, one to display the main menu and one that will execute operations in sequence. Later the separate programs will be integrated, each becoming a Java method.

3. When you integrate your main-menu program with your implementation of the five operations, write a sub-menu under the Update option, allowing the user to choose which attribute of the item to update.

4. The code should then be integrated together to be in *modular* form, with method calls for each menu option and a method definition for each module. Be sure that each method is properly documented at the top of the definition.

5. *Test results* must show that the code works correctly for all menu options and a variety of input data.

6. Final submission should include each version, including a version that operates on one contact using several programs, a one-contact version that has one program with multiple methods, and a version that manages a collection of contacts. The final submission is to be separated into sections on *specification*, *design*, *code*, and *testing*. Use dividers in a thin (not loose-leaf) binder. Submit your source code and a sample data file.

### Recipe for one way to organize the semester project

There are many correct ways to do this project. Below is one plan.

1. Restate the project specs in your own words and make them a comment at the start of your program file.
2. Read steps 3-5 below and summarize them in a design documentation comment.
3. Declare a web-page structure type or class, with members as described in the project spec.
4. Declare a web-page collection structure type or class, with an array of web-page items and an integer (number of web pages in the collection) as members.
5. Create a menu that lets user choose to input one new web page, display the collection, save the collection to file, or read a collection from file.
6. Test your menu and functions.
7. Add searching and sorting features as able to do so based on classroom discussion and handouts.

### Requirements for graded assignments

- Each submission must include a description of the *process* by which you did the work, such as what difficulties you had, what new skills you learned, what new concepts became clearer, what work if any is not completed, etc. One part of process documentation can be *muddy concepts* feedback, in which you describe what is least clear in the classroom presentation and textbook.
- Code, documentation, and input/output should all be clear and readable; presentation counts.
- Comments should express *clearly* what the program does. Document program's name and purpose at top in a comment; document any function definitions similarly. Place your name, the date, and the assignment and problem numbers at the top of each source file.
- Use a standard indenting scheme.
- Homework *must* be submitted on time for credit. "It is *strongly* recommended that you start very early on each homework assignment and *ask questions* if any part is unclear or seems difficult" (syllabus).
- You are responsible for submitting only your own work and keeping your own work to yourself so that others will be able to work independently. Work with copied code or comments will be returned ungraded.

### Grading criteria for final submissions

Specification documentation	15%
Design documentation	15
Presentation	10
Data definition	15
Control structures	15
Modules	15
Testing	15

### Notes

All students are to work independently and submit their own code and documentation.

The project is designed to be done in stages. It is not acceptable to skip these stages and submit only the entire project at the end of the semester.

You may choose to add features to the above specification. Be sure to document these.

## Group work on topic 1 (JavaScript)

Post all group-work solutions as replies to the problem description posting, with a subject line that specifies the problem solved and lists names of group members who participated in solving the problem.

1. Write and test programs in JavaScript whose inputs and outputs are as specified below. The program should be embedded in an HTML file. Be sure to comment the program thoroughly, including the HTML file name.

<i>input</i>	<i>output</i>
(a) two numbers	their sum
(b) one number, $x$	$x^3$
(c) two numbers, $a$ and $b$	$a^2 + b$
(d) two numbers	their product
(e) two numbers	their quotient

2. See example file *count-yes.htm*. Modify it to create a web page that works as follows: Four buttons are displayed: “\$1”, “\$5”, “\$10”, “Done”. The user presses the 1, 5, and 10 buttons each a certain number of times, to request these amounts. After pressing “Done,” the user sees a number reflecting total amount requested. To solve this problem, you need to use an expression that assigns to *OnClick* a string that starts with 'alert(', ends with ')', and contains an expression that computes the sum of the \$1 amounts, \$5 amounts, and \$10 amounts.

3. See the newly written last section of the JavaScript handout and the example, *power.htm*, which uses the *while* statement in JavaScript. The following problems may be solved by modifying *power.htm*.
  - (a) Prompt for an input  $x$  and output the sum of all numbers from 1 to  $x$ .
  - (b) Prompt for an input and output its factorial.
  - (c) Prompt for an input and output the integer part of its base-2 logarithm, i.e., the number of times the number can be divided by two before reaching a value less than one.

## Group work 2 (Hardware)

### I. Binary conversion

Convert to decimal, showing your work:

- (1) 10110011
- (2) 10101110
- (3) 11011011
- (4) 10101110
- (5) 10110010
- (6) 01010011
- (7) 01100101
- (8) 10011101
- (9) 11001111

A groups should solve the problem with the same number as the group. Post solutions, as replies to this post, with names of group members who participated.

### II. ASM testing

Download the program *asm\_setup.exe* and use it to install the program *asm.exe* in your student account. Run *asm.exe*, and use it to step through the program *xy.asm* (available in Discussion Board, topic 2), choosing step mode. To open *xy.asm* within *asm.exe*, use File / Open. Test the program for two or more different input values.

- (a) From your observation, what occurs when the fifth line of the program, *store y*, executes?
- (b) Test the program for a few input values and post your test file, which will be 'xy.out'.
- (c) Based on your observation, write a formula that accurately describes the relationship between input and output of program *xy.asm*. (Your formula could be of the form "Output is  $n$  larger than input," or "Output is random," or "Output is same as input.")
- (d) Suggest a name for the program that describes what it does better than the name *xy.asm*, and better names for the data labels  $x$  and  $y$ .

### III. ASM programming

1. These are a little tricky. Each group take the question with the same number as the group (1 Fatima-Monelle, 2 Reb-Erica, 3 Nick, 4 Chady, 5 James-Ryan, 6 Shane, 7 Chris-Chris-PJ, 8 Josh-Patrick-Daravann, 9 Pat-Jason, 10 Daron-Kelly-Derek, 11 Gwen). For groups with numbers over 9, wrap around to 1, 2, etc.

2. For each expression below, write and test a program for ASM that inputs value  $x$  (or values  $x_1$ ,  $x_2$ , etc.) and outputs the expression's value. *Hint*: for multiplication and division, use repeated addition or subtraction.

3. Post solutions as attachments to replies to this post. Also post test results, which will be in files named the same as the program files, except ".out" replaces ".asm".

1.  $x_1 + x_2 - (2x_3)$
2.  $x_1 - 2x_2$
3.  $2x_1 + x_2$
4.  $3x_1 - 2x_2$
5.  $2(x_1 + x_2)$
6.  $2(x_1 - x_2)$
7.  $x_1 + x_2 + x_3$
8.  $x_1 + x_2 + x_3 - x_4$
9.  $2(x_1 + x_2 + x_3)$

### IV. ASM challenge problems

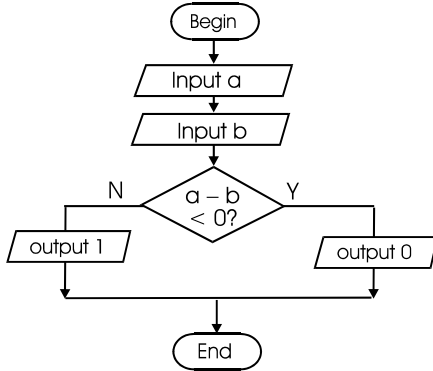
1. Solutions for the following problems cannot be tested on the current version of ASM, which does not correctly implement conditional jumps. But students in this class have solved them in their heads without testing. Do so using pencil and paper. *Suggestion*: see the example in the slides, 'count.asm', and work from that, noting how the 'jump', 'jump0' or 'jump-' instructions are used.

1.  $20 * x_1$
2.  $x_1 * x_2$
3.  $x^2$
4.  $x_1 / x_2$  (integer quotient)
5. remainder after computing quotient  $x_1 / x_2$
6. 0 if  $x$  is divisible by 2, 1 otherwise
7. The sum of all numbers from 1 to  $x$
8. The largest of 5 inputs
9. The smallest of 5 inputs

## Group work 3 (Program design)

- A. See semester project specification sheet.
- Use your own words to explain the specifications of the project; specifically, what are the inputs, what are the outputs, and what is the relationship of inputs to outputs.
  - Write a flowchart of the main-menu loop described in the specification.
  - Write a module hierarchy for the application. Describe the modules needed to implement the specifications (#1 above), considering the menu items as modules.

B. Consider the flowchart below.

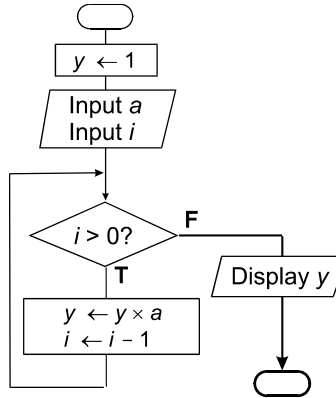


For each of the input pairs  $(a, b)$ , shown below, show the resulting output.

	<b>Input a</b>	<b>Input b</b>	<b>Output</b>
1.	2	1	_____
2.	1	3	_____
3.	4	2	_____
4.	7	10	_____
5.	3	2	_____
6.	3	5	_____
7.	6	2	_____

Informally, what does the output of this algorithm tell about the two inputs?

- C. Use the table below to trace the algorithm specified in the flowchart below for the given inputs of  $a$  and  $i$ :

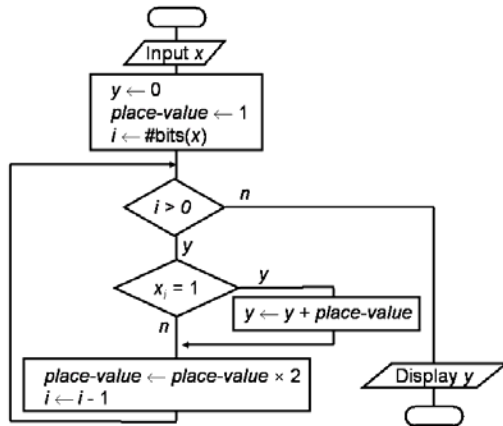


- 2 and 3;
- 4 and 2;
- 5 and 1;
- 10 and 3;
- 12 and 2;
- 1 and 6;
- 2 and 4.

<b>a</b>	<b>i</b>	<b>y</b>	<b>output</b>
_____	_____	_____	_____
_____	_____	_____	_____
_____	_____	_____	_____
_____	_____	_____	_____
_____	_____	_____	_____

D. Show a trace of this flowchart of binary-to-decimal conversion, for inputs

- (a) 11011;
- (b) 101101;
- (c) 101010;
- (d) 011001;
- (e) 010110;
- (f) 100101;
- (g) 100111



- E. Using flowchart or pseudocode, design an algorithm or interactive process that
1. loops to accept four numbers and displays their sum
  2. accepts two numbers and displays the larger.
  3. accepts three numbers and displays the largest;
  4. four numbers;
  5. five numbers
  6. prompts for two integers and displays their quotient, using only subtraction; show an error message if the divisor is 0.
  7. accepts a number  $x$  and displays the sum of all the whole numbers from 1 to  $x$ ;
  8. accepts a number  $x$  and displays the product of all the whole numbers from 1 to  $x$
  9. prompts for two integers and display their product, using only addition and subtraction operations in your calculations;
  10. prompts for a string and tells whether it contains any doubled-up characters.

- F. See the group work for topic 2 (Hardware), problem. Use flowchart or pseudocode to design a solution to one of the problems, using only the arithmetic operations supported by the ASM processor simulator, i.e., addition and subtraction. This means writing a loop for multiplication and division.
1.  $20 * x_1$
  2.  $x_1 * x_2$
  3.  $x^2$
  4.  $x_1 / x_2$  (integer quotient)
  5. remainder after computing quotient  $x_1 / x_2$
  6. 0 if  $x$  is divisible by 2, 1 otherwise
  7. The sum of all numbers from 1 to  $x$
  8. The largest of 5 inputs
  9. The smallest of 5 input

## Group work 4 (Java basics)

*For all problems, comment your code with names of students who contributed significantly to the solution, and show test results.*

### A. Introductory Java assignment

#### Steps

1. Download the program *add.java* (Discussion Board, Topic 4, Examples), and compile and run it so that you are sure you know how to use the compiler (see handout, “Using Java Development Tools”).
2. Edit the program so that it inputs values to variables as indicated below and outputs the expression below that has the same number as your group. (Name the variables as you please.) Save the program under a new file name.
3. Comment it so that it displays your group members' names at the top and so that it explains your code in your own words.
4. Test the program with a couple of different sets of inputs, saving test results as a file. Text can be copied from the DOS window by expanding the command-line window (*Alt-Enter*) and pressing *PrtScr*.
5. Post solutions as replies to this post or attachments. Also post test results (you may include test results as a long comment at the end of your program.)

*Problems* (each group does the problem that matches the group's number):

1.  $x_1x_2 - 2x_3$
2.  $x_1 + 2x_2 - x_3$
3.  $2x_1 + x_2 + x_3$
4.  $3x_1 - 2x_2$
5.  $2(x_1 + x_2) + x_3$
6.  $x_1 + 2(x_2 - x_3)$
7.  $x_1 + x_2 + x_3$
8.  $x_1 + x_2 + x_3 - x_4$
9.  $2(x_1 + x_2 + x_3)$

### B. Output of text

Write initials of group members in large letters vertically (see Horstmann, p. 178, Ex. P4.15).

*Challenge:* Write to a text file.

### C. Numeric types

Write a program to help you find the approximate largest numbers represented (without overflow) by one of the following types. One way to solve this problem would be to take an input value into a variable of the specified type, and see what the value output is. If the value output is wrong, then overflow has occurred.

1. *byte*
2. *short*
3. *int*
4. *long*
5. *float*
6. *double*

You may need to use several statements or several runs of your program to help home in on the highest value. See the textbook for guidance on numeric types and overflow.

### D. Algebra to Java conversions

Horstmann, p. 171, Ex. R4.1-4.3.

### E. Computing values on two inputs (optional)

Horstmann, p. 174, Ex. P4.4 – no classes required.

### F. Graphics (optional)

Write a program to draw simple stick-figure pictures of the group members, labeled with their names. See Horstmann graphics chapter subsections for guidelines.

## Group work 5 (Control structures)

Post solutions on the Discussion Board as replies to the threads for the problems.

### A. Syntax errors

Horstmann, p. 217, Ex. R5.1. Each group do one problem, a=1, b=2, etc.

### B. Writing loops in Java

Horstmann, pp. 221-223, Ex. P5.1-P5.10.

### C. Converting flowcharts to Java code

See topic 3 (Design), group work problems E and F. Convert to Java the flowcharts you wrote for your group's problems.

### D. Numeric conversions between bases

Implement conversion algorithms in Java of the kind below that corresponds to your group number:

1. Binary to decimal
2. Decimal to binary
3. Hexadecimal to binary
4. Binary to hexadecimal
5. Hexadecimal to decimal
6. Decimal to hexadecimal

### E. String manipulation

1. Search an input string for an input character, display the character in brackets within the string if it is found, otherwise display "not found in " and the string.
2. Find the longest run of the same character in input string, display it in brackets.
3. Find the last period in an input string, bracket it in output.
4. Input a name in two or more strings, display "Hello " and the first name only.
5. Input a string and display a count of the number of vowels found.
6. Input a string and upper-case its characters for output.
7. Check an input string and tell whether it is a palindrome (spells same way backwards as forwards).

*As specified on the course syllabus, document all code with the name and purpose of the program, your name, the date, and the homework number; document process. Use meaningful variable names.*

1. Write a program that displays a menu, as indicated on the semester project specification sheet. Menu should display a number or letter beside each menu option, input into a variable a number or letter corresponding to the user's choice, and echo (display) the choice token. (E.g., "You chose 1.")