Software Engineering 종합문제은행

Part I. Definition & Introduction of basic Software Engineering

Please answer the following questions:

* Software Engineering
- What is software? (vs. hardware)
- What is software engineering? Why is it important?
- What are Key challenges of software engineering? How to overcome them?
- What are different from computer science and management science?

* What is Code of ethics

* Software Process model: What, Why, How? and compare one with another (pros & cons)
- waterfall model
- Incremental development process
- Reuse-oriented software engineering
- prototype-based development process
- Boehm's Spiral model
- plan-driven vs. agile development process model: define and compare
- Extreme programming

* software development activities
- software specification
- software design and implementation
- validation and testing
- software/system evolution

* RUP (Rational Unified Process)

* Requirements Engineering (RE)
- What is RE?
- Why is RE important? What if RE is wrong?
- What are Key challenges of RE? How to overcome them?
- What are five key failure factors related to RE among CHAOS projects (See the WinWin slides)
- What are functional and non-functional requirements? Why are non-functional requirements hard to elicit?
- How to validate requirements? How do you know whether your requirements are correct and whether your requirements are correctly described?
- What are types of requirements specification? What are advantages and disadvantages of them?
- How to represent requirements consistently, completely, and non-ambiguously? How does the software industry do for this goal?

* System Modelling: what, why, how of the following concepts?
- External perspective, Behavioral perspective, Structural perspective

- Context Models
- Behavioral Models: Data-Flow models, state machine models
- Data Models
- UML (Object) Models: aggregation and inheritance, class diagram, sequence diagram, collaboration diagram, state diagram (see Chapter 5 & 7)

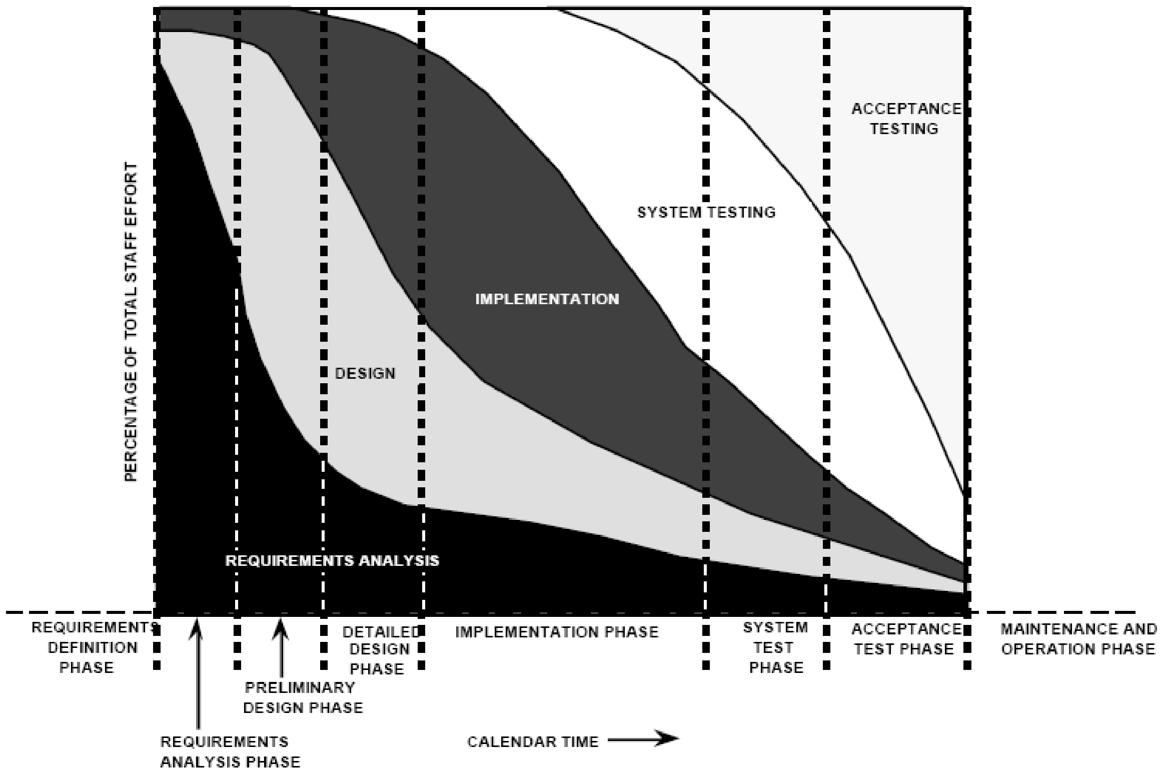
* Design pattern
- What is design pattern?
- Why is design pattern important? (benefits)
- Explain design patters using examples (e.g., MVC: Model-View-Controller, Inversion of control in frameworks)

* Describe What, Why, and How about the following concepts:
- Validation
- Verification
- defect testing
- V model
- software inspection
- faults vs. defects
- automated static analysis
    > control flow analysis
    > data use analysis
    > interface analysis
    > information flow analysis
    > path analysis

* Describe What, Why, and How about the following concepts:
- integration testing
- release testing
- performance testing
- component testing
- interface testing
- partition testing
- structural testing
- path testing
- test automation

Part II. Application Question of Software Engineering

1. Diagram shown above illustrates how "typical" waterfall-based software development project may proceed. You have been asked to brief some of the key software engineering principles using the diagram to clients who are not experts on the subject. What are they? Explain as many (up to three) of key software engineering principles as you can. Be detailed in your explanation.



2. Project management has to deal with finding the right balance between schedule, cost, and quality. Please suggest your proposed formula/rules/principles to optimize among schedule, cost, and quality factors. List top three risks of your project if your project loses the balance of three factors and discuss how to mitigate the risk (risk mitigation plan).

Hints: Schedule, cost and quality have trade-off relationship. That is, if you want to improve better quality, schedule could increase or/and cost could increase. Or, if you want to finish the project as soon as possible (decrease schedule), you could pay more (higher cost) or may have more errors (lower quality). Your project process model (e.g., Agile or Waterfall model) may suggest you the different balance point among the three factors.

3. Software Engineering is a study of principles and practices of maximize added-values with minimum resources. What did you create added-values with software engineering in your project (compare it with your previous projects experience without software engineering in different classes)? What are your lessons learned in your project and how to improve them if you do the same project next time?

4. (a) Draw an activity network based on the following table of task durations and dependencies. Note that the starting date is 1/1/02 (MM/DD/YY). Assume that every day is working day including Saturday and Sunday.

Task	Duration (calendar days)	Dependencies
T1	7	
T2	14	
T3	14	T1 (M1)
T6	7	T1, T2 (M3)

(b) List the critical path in the activity network that you drew above (5 pts).

5. A typical formula for the COCOMO model is "Effort = A * Size^B * M". Please calculate the following two person-months (two boxes in Figure 23.10 below). Hint: You may want to drive A, B, and M using the information shown below.

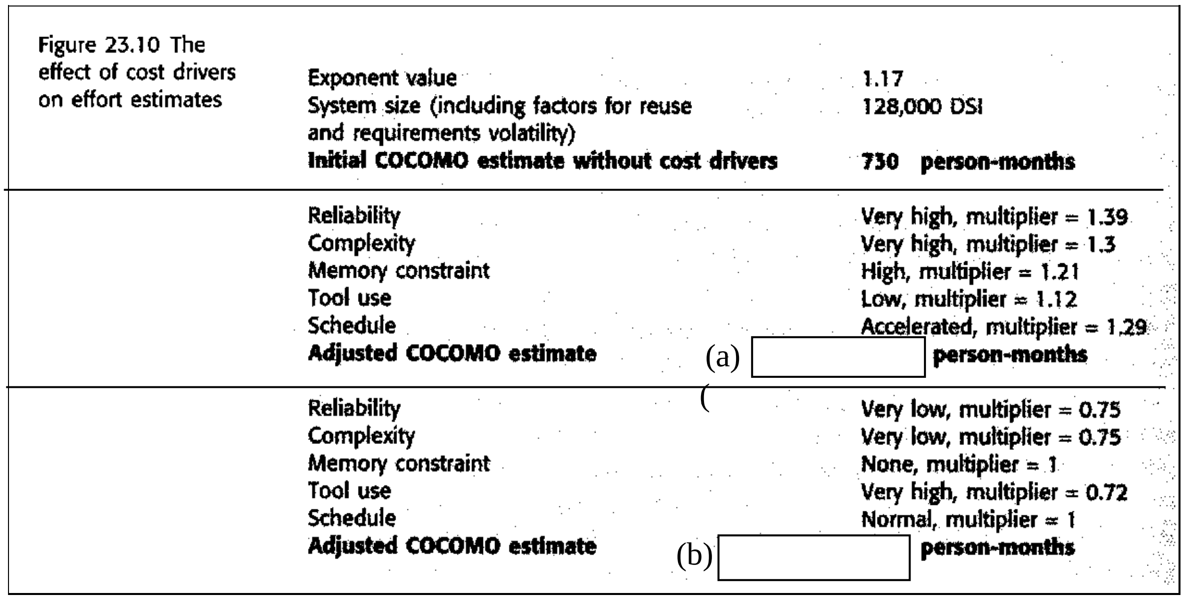
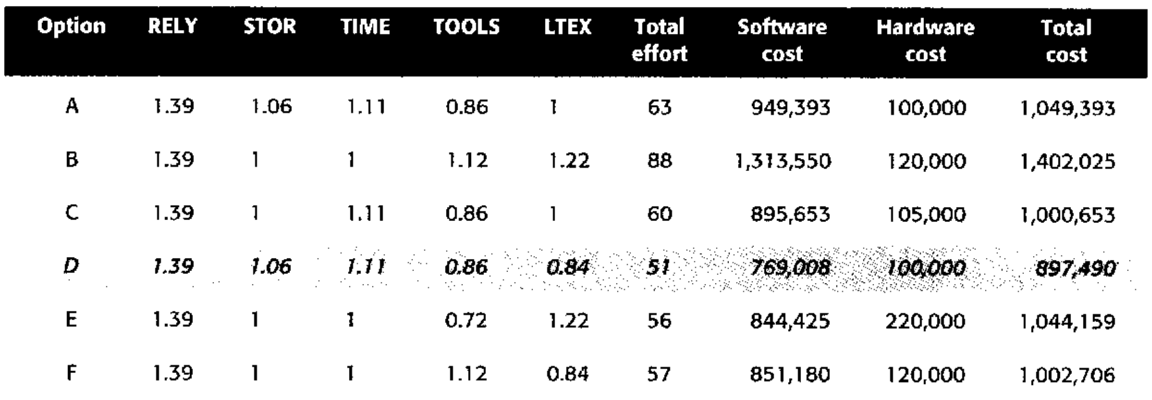


Figure 23.10 The effect of cost drivers on effort estimates

Exponent value	1.17
System size (including factors for reuse and requirements volatility)	128,000 DSI
Initial COCOMO estimate without cost drivers	**730 person-months**
Reliability	Very high, multiplier = 1.39
Complexity	Very high, multiplier = 1.3
Memory constraint	High, multiplier = 1.21
Tool use	Low, multiplier = 1.12
Schedule	Accelerated, multiplier = 1.29
Adjusted COCOMO estimate	(a) $\boxed{}$ **person-months**
Reliability	Very low, multiplier = 0.75
Complexity	Very low, multiplier = 0.75
Memory constraint	None, multiplier = 1
Tool use	Very high, multiplier = 0.72
Schedule	Normal, multiplier = 1
Adjusted COCOMO estimate	(b) $\boxed{}$ **person-months**

6. The following table shows the hardware, software and total costs for the options A-F. Applying the COCOMO 2 model without cost drivers predicts an effort of 45 person-months to develop an embedded software system. The average cost for one person-month of effort is $15,000. Option A represents the cost of building the system with existing support and staff and hardware ('baseline'). Option B represents that upgrading hardware does not necessarily reduce cost because the experience multiplier (LTEX) is more significant. Please answer the following questions.

    (a) What does Option C represent? (3 pts)
    (b) What does Option D represent? (3 pts)
    (c) Which option (Option C or D) would conservative project managers probably select? Why? (4 pts)



Option	RELY	STOR	TIME	TOOLS	LTEX	Total effort	Software cost	Hardware cost	Total cost
A	1.39	1.06	1.11	0.86	1	63	949,393	100,000	1,049,393
B	1.39	1	1	1.12	1.22	88	1,313,550	120,000	1,402,025
C	1.39	1	1.11	0.86	1	60	895,653	105,000	1,000,653
D	*1.39*	*1.06*	*1.11*	*0.86*	*0.84*	*51*	*769,008*	*100,000*	*897,490*
E	1.39	1	1	0.72	1.22	56	844,425	220,000	1,044,159
F	1.39	1	1	1.12	0.84	57	851,180	120,000	1,002,706

7. [UML and Object Oriented Design] The following description is an example of a library system.

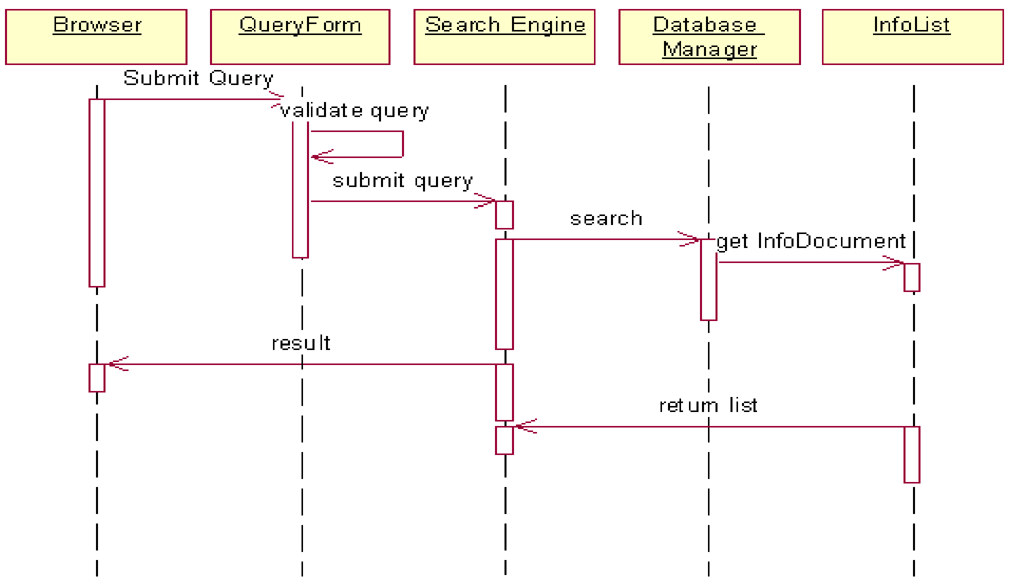
Description of a Library System

Books and journals. The library contains books and journals. It may have several copies of a given book. Some of the books are for short term loans only. All other books may be borrowed by any library member for three weeks. Members of the library can normally borrow up to six items at a time, but members of staff may borrow up to 12 items at one time. Only members of staff may borrow journals

Borrowing. The system must keep track of when books and journals are borrowed and returned, enforcing the rules described above.

(a) Please draw a class diagram of the library system based on the above description (please consider inheritance, associations, multiplicity, etc. However, do not consider attributes and operations). You can use a grammatical analysis of a natural language description of a system to identify objects and classes. Objects and attributes are nouns.
(b) Another approach to identify objects is a scenario-based analysis. Please develop a simple scenario based on the above description and identify objects in the scenario.

8. Transform the following sequence diagram into a collaboration diagram with interactions in UML. Please determine whether this sequence diagram is logically right or not. If not, please correct it.



9. Testing

(a) Create 8 test cases for binary search routine (see below) based on equivalence partitions. The test cases should be composed of Input Array (T), Key (Key), and Output (Found, L). The following explanation would be helpful (12 pts). ***Please assume that INPUT array (T) has a range of the following values: 1000 to 2000. Do Not Copy Test Cases shown in the Textbook.***

Structure testing is an approach to testing where the tests are derived from knowledge of the software's structure and implementation. Structural testing is usually applied to relatively small program units such as subroutines or the operations associated with an object. As the name implies, the tester can analyze the code and use knowledge about the structure of a component to derive test data. The analysis of the code can be used to find how many test cases are needed to guarantee that all of the statements in the program or component are executed at least once during the testing process. Figure 20.9 is a binary search routine. A pre-condition is to use an ordered array, that the value of the lower bound of the array must be less than the value of the upper bound. By examining the code of the search routine, we can see that binary searching involves splitting the search space into three parts. Each of these parts makes up an equivalence partition. Test cases where the key lies at the boundaries of each of these partitions should be chosen to exercise the code.

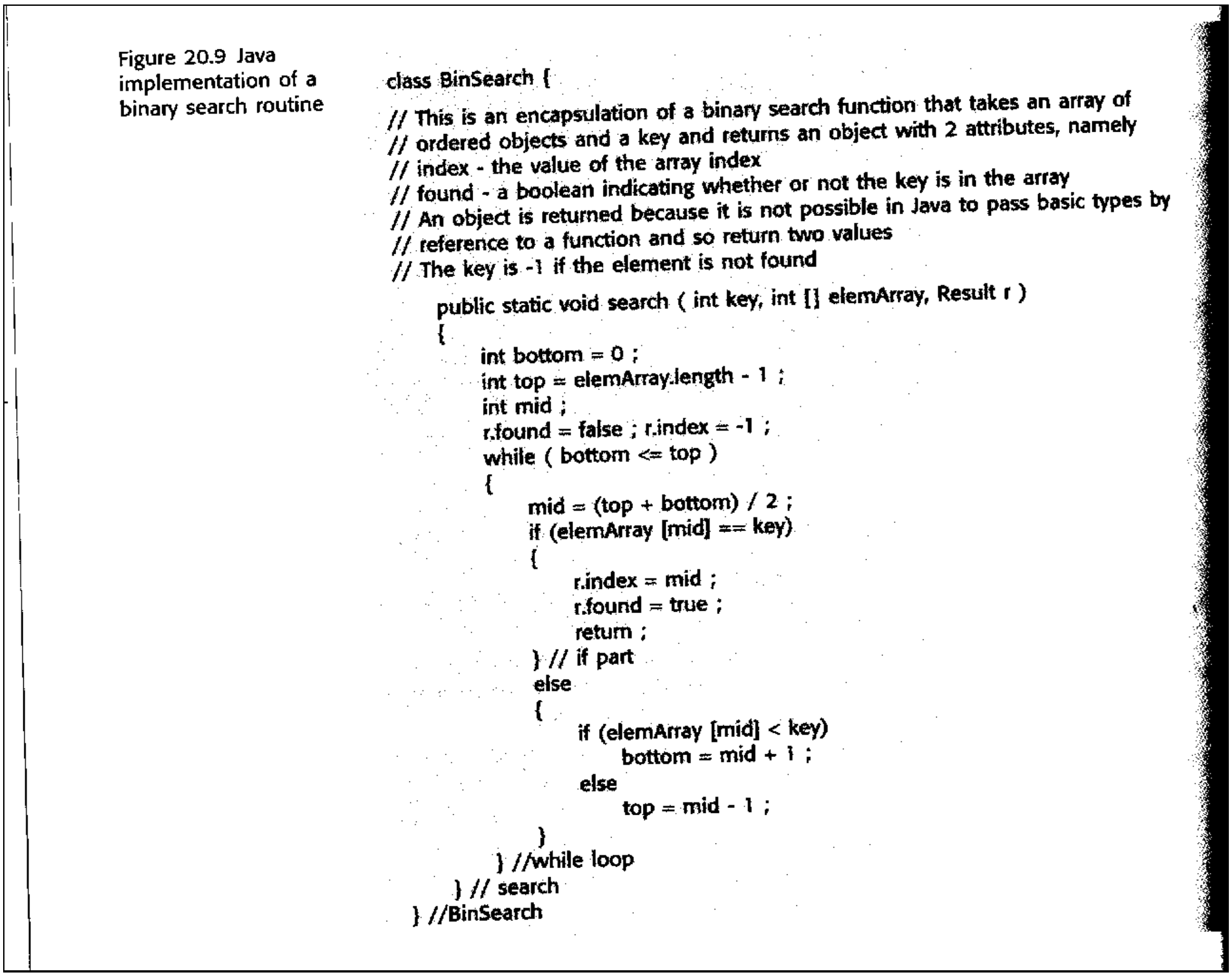


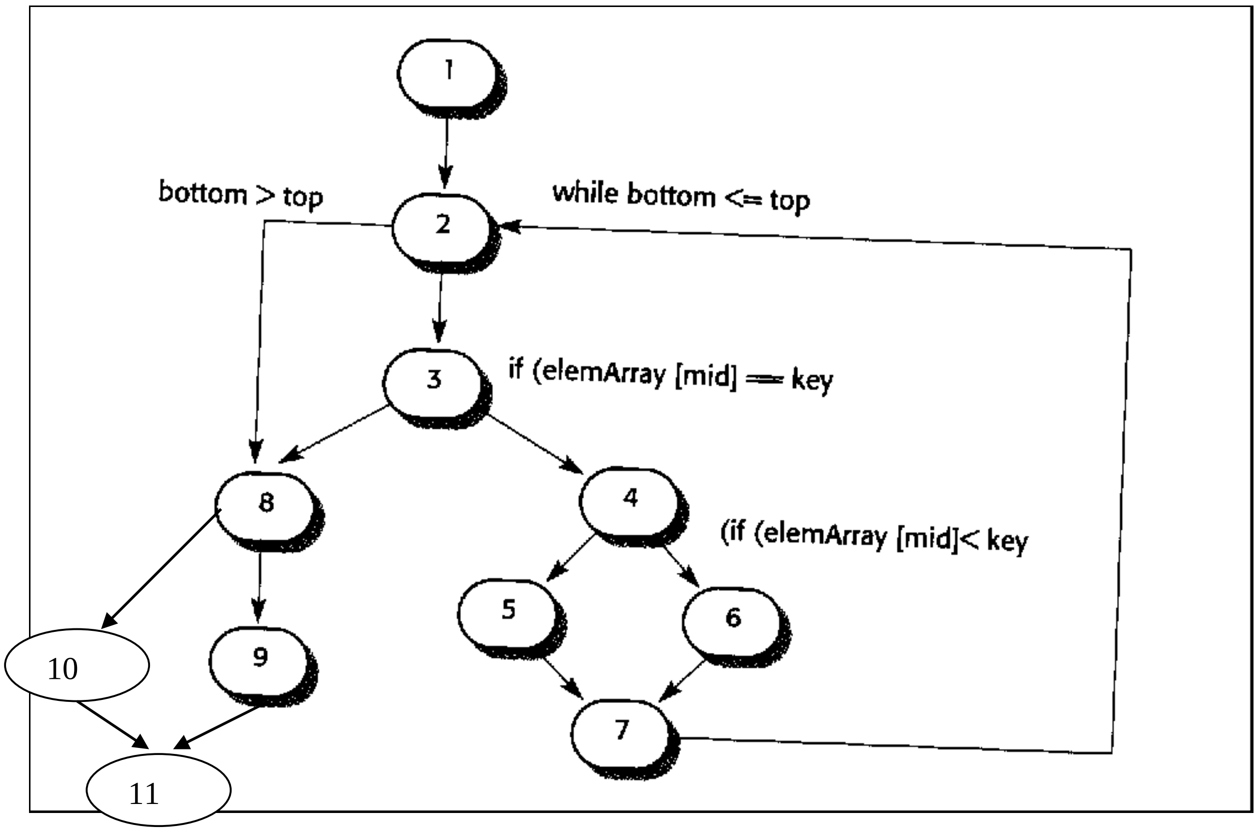
Figure 20.9 Java implementation of a binary search routine

```
class BinSearch {

// This is an encapsulation of a binary search function that takes an array of
// ordered objects and a key and returns an object with 2 attributes, namely
// index - the value of the array index
// found - a boolean indicating whether or not the key is in the array
// An object is returned because it is not possible in Java to pass basic types by
// reference to a function and so return two values
// The key is -1 if the element is not found

    public static void search ( int key, int [] elemArray, Result r )
    {
        int bottom = 0 ;
        int top = elemArray.length - 1 ;
        int mid ;
        r.found = false ; r.index = -1 ;
        while ( bottom <= top )
        {
            mid = (top + bottom) / 2 ;
            if (elemArray [mid] == key)
            {
                r.index = mid ;
                r.found = true ;
                return ;
            } // if part
            else
            {
                if (elemArray [mid] < key)
                    bottom = mid + 1 ;
                else
                    top = mid - 1 ;
            }
        } //while loop
    } // search
  } //BinSearch
```

(b) The following diagram is the flow graph for the ***modified*** binary search procedure. Please provide four independent paths through the binary search flow graphs (4 pts). And calculate the cyclomatic complexity of the search flow graphs (4 pts).

Note: If all of these paths (of the flow graphs) are executed we can be sure that:

1. every statement in the method has been executed at least once, and
2. every branch has been exercised for true and false conditions.


1
2
bottom > top
while bottom <= top
3
if (elemArray [mid] == key
8
4
(if (elemArray [mid]< key
5
6
10
9
7
11