NAME: Kenneth Robinson

CHAPTER: 7

LAB: Lab 6

ANIMATED FLASHCARDS

1. Assembler

2. Assembly language

3. Code-coverage (clear-box) testing

4. Comment

5. Data-coverage (black-box) testing

6. Loader

7. Software piracy

8. Test plan

9. Test-plan implementation

10. Unary operator

11. Virtual computer

12. Machine language

BOOK EXERCISES

7. C

8. D

9. A

10. B

11. A

12. B

13. D

14. E

15. C

16. A

17. B

18. A

19. A

20. A

21. B

22. B

23. A

24. B

25. A

26. A

27. B

28. B

29. A

30. B

31. B

32. A

33. A

34. B

35. A

36. A

37. Understanding the problem, devising a plan, carrying out the plan, and looking back.

42. An algorithm is a set of instructions for solving a problem in a finite amount of time using a finite amount of data.

44. Algorithm development phase, implementation phase, and maintenance phase.

45. In Polya's list, the human executes the plan and evaluates the results. In a computer solution, a program is written that expresses the plan in a language that the computer can execute. The human then takes the computer output and evaluates the results.

51. The top-down design process is characterized by successive layers of refinement. The top-level tasks are listed. At each succeeding level, the tasks from the previous one are further developed and expanded.

52. An abstract step is which further development is needed. A concrete step is which all the steps are fully specified.

56. Information is any knowledge that can be communicated. When information is in the form that a computer can use, it is called data. Thus data is any knowledge that can be communicated in a form that a computer can process.

58. a. Information hiding is important because defers details until the level where the details are important. This process keeps an algorithm from being dependent on the implementation details, which may change.

b. Three examples of information hiding are driving a car, talking on the telephone, and turning on the television.

66. a. 11 b. 5 c. 11 d. 11 e. 11

67. a. 4 b. 4 c. 1 d. 4 e. 4

LAB EXERCISES

**EXERCISE 1**

1. Estimated distance to be traveled, how many people, number attractions going to be visited, the price per attraction, the current price of gas, the duration of the trip, and the number of people traveling.
2. The solution would be an estimate of how much money the family would need in order to complete the trip.   
   The output would be an estimate of how much money it would be necessary to go on the trip and
3. I think there are several solutions. For larger families, they might want to be able to allocate multiple hotel rooms, and larger vehicle costs. Also, some families may want to rent a vehicle instead of driving their own.

Ask for a starting city and destination. Then find routes between them with associated costs. Choose the best routes.

Find destinations that will accommodate all family members.

Find hotels along the way that offer the best value.

1. The things that a present-day computer can do are: find a highway route from Philadelphia to New York, add the hotel cost to the running total of how much money has been spent, sort the list of cities by priority number, print a list of cities in the order in which they should be visited, and look up the distance from city A to city B.
2. While city1 != city2 and city1 != null and city2 != null and distance = found then  
    look up city distances from table  
    if distances do not exist  
    then generate a route using known distances for other cities between  
    end if   
   end while

The pitfalls in the loop would be accurately determining the distance between the current city and the next city. Also, it may suggest taking a route that is a longer distance, but has a shorter travel time.

The algorithm can be tested by using Mapquest, Google Maps, or any other GPS navigation device or software.

**EXERCISE 2**

1. The class needs to keep track of the following: album name, artist, release year, and titles of each song.  
  
Create a **CD catalog** that includes all of my CDs with **information** about what music is on them, including **artist**, **album title**, the **year** the album was released, and the title of each **song** on the album.

3.

|  |  |  |  |
| --- | --- | --- | --- |
| Class Name: *Album* | Superclass: | | Subclass: |
| Responsibilities | | Collaborations | |
| *Initialize self (artist, tracks, year, album title)* | | *Tracks, String* | |
| *Print* | | *Tracks, String* | |
| *getArtist* | | *String* | |
| *getTracks* | | *Tracks, String* | |
| *getYear* | | *Int* | |
| *getAlbumTitle* | | *string* | |

|  |  |  |  |
| --- | --- | --- | --- |
| Class Name: *Tracks* | Superclass: | | Subclass: |
| Responsibilities | | Collaborations | |
| *Initialize self (Track title)* | | *String* | |
| *Print* | | *String* | |
| *getTrackTitle* | | *String* | |

|  |  |  |  |
| --- | --- | --- | --- |
| Class Name: *SortedList* | Superclass: | | Subclass: |
| Responsibilities | | Collaborations | |
| *Insert (Album)* | | *Album* | |
| *Print itself* | | *Album* | |

4. Containment

Inheritance

Collaboration

5. containment songs album

Inheritance title artist

collaboration

6a. In the album class, the getYear can be omitted as long as the year of the album is omitted.

6b. Album is a collection of songs. If one of the parts leaves, such as title or the artist, then it is not an album. A combination of all these things makes the final product.

6c. Determines the number of tracks that are included in an album.