软件工程部分知识点归纳及练习

一、知识点

1. 软件的复杂性是固有的，它引起人员通信困难、开发费用超支、开发时间超时等问题。

2. 螺旋模型是在瀑布模型和增量模型的基础上增加了风险分析活动，关键不足在于不能适应需求的动态变更。

3. 软件不只是用程序设计语言（如 PASCAL ,C,VISUAL BASIC 等）编写的程序，编写程序代码只是软件开发的一个部分。

4. 快速原型模型可以有效地适应用户需求的动态变化。

5. 生产高质量的软件产品是软件工程的首要目标。

6. 软件开发人员对用户需求的理解有偏差,这将导致软件产品与用户的需求不一致,是产生软件危机的一个原因。

7. 开发一个软件项目总存在某些不确定性,即存在风险.有些风险如果控制得不好,可能导致灾难性的后果。

8. 缺乏处理大型软件项目的经验，是产生软件危机的一个原因。

9. 瀑布模型本质上是一种线性顺序模型，增量模型本质上是一种快速原型模型。

10. 可行性分析是在系统开发的早期所做的一项重要的论证工作,它是决定该系统是否开 发的决策依据,因必须给出可行或不可行的回答。

11. 软件生存周期模型包括瀑布模型、快速原型模型、增量模型、螺旋模型、喷泉模型等。

12. 软件是一种逻辑产品。

13. 可行性研究要进行一次简化的,压缩了的需求分析.

14. “软件危机”是指软件开发和维护中出现的一系列问题，成本高、生产率低、质量得不到保证、需求不能充分理解都是是软件危机的表现形式。

15. 原型化方法是用户和设计者之间执行的一种交互过程,适用于需求不确定性高的系统。

16. 软件工程中的各种方法是完成软件工程项目的技术手段,它们支持软件工程的各个阶段.

17. 研究开发所需要的成本和资源是属于可行性研究中的经济可行性研究的一方面。

18. 快速原型模型的主要特点之一是及早提供工作软件。

19. 软件生存周期模型是描述软件开发过程中各种活动如何执行的模型.

20. 软件工程由方法,工具和过程三部分组成,称软件工程的三要素.

21. 基于计算机系统的软件要素中的软部件由程序,数据和文档组成.

22. 瀑布模型是以文档为驱动、适合于软件需求明确的软件项目的模型。

23. 螺旋模型是风险驱动的，而瀑布模型是文档驱动的。

24. 从事物的组成部件及每个部件的属性,功能来认识事物.这种方法被称为面向对象的方法。

25. 从事物的属性侧面来描述事物的方法就是面向数据的方法。

26. 面向对象(Object Oriented)方法是将现实世界的事物以对象的方式映射到计算机世界的方法。

27. 软件工程的 7 条基本原理：1、用分阶段的生命周期计划严格管理；2、坚持进行阶段评审；3、实行严格的产品控制；4、采用现代程序设计技术；5、结果应能清楚；6、开发小组的人员应该少而精 7、承认不断改进软件工程实践的必要性。

28. 软件需求是指用户对目标软件系统在功能,性能,行为,设计约束等方面的期望.

29. 数据流图就是用来刻画数据流和转换的信息系统建模技术。

30. 用户对软件需求的描述不精确,往往是产生软件危机的原因之一。

31. 需求分析阶段的成果主要是需求规格说明,该成果与软件设计,编码,测试直至维护都有较大关系。

32. 软件需求是指用户对目标软件系统在功能,性能,行为,设计约束等方面的期望。

33. 需求规格说明书是需求分析阶段最重要的技术文档之一

34. 需求分析最终结果是产生需求规格说明书。

35. DFD 中的每个加工至少需要一个输入流和一个输出流。

36. 需求分析阶段的任务是确定软件系统的功能

37. 需求分析的任务不包括系统设计。

38. 需求规格说明书的作用包括：作为软件验收的依据、用户与开发人员对软件要做什么的共同理解、作为软件设计的依据等

39. 在结构化分析方法中，用以表达系统内数据的运动情况的工具有数据流图。

40. 结构化分析方法（ＳＡ）是一种面向数据流的需求分析方法.

41. 验证软件需求正确性的四个方面包括一致性、完整性、现实性、有效性。

42. 在面向对象软件开发方法中，类与类之间主要有继承和聚集的关系。

43. 面向对象的特征主要包括多态性、继承性、封装性。

44. 软件开发过程中,抽取和整理用户需求并建立问题域精确模型的过程叫面向对象的分析。

45. 用户界面的可使用性是用户界面设计最重要的也是最基本的目标。

46. 软件概要设计的主要任务就是软件结构的设计,面向数据流的设计方法是将数据流映射成软件结构。

47. 软件模块之间的耦合性越弱越好。

48. 模块化，信息隐藏，抽象和逐步求精的软件设计原则有助于得到高内聚，低耦合度的软件产品。

49. 内聚度标志一个模块内部各成分彼此结合的紧密程度,按其高低程度可分为七级,内聚度越高越好。

50. 在模块耦合性类型中，模块之间独立性最差的类型是内容耦合，在实际编程时一定要避免出现，为了提高模块的独立性，模块之间最好是数据耦合。模块耦合越弱，则说明模块的独立性强。

51. 尽量用数据耦合，少用控制耦合，限制公共环境耦合的范围，完全不用内容耦合。

52. 在软件结构图中，扇入数大说明该模块的重用率高。

53. 为了提高模块的独立性,模块内部最好是功能内聚，模块的内聚性最高的是功能内聚。

54. 软件结构图中,模块框之间若有直线连接,表示它们之间存在调用关系一个软

件的宽度是指其控制的跨度，一个软件的深度是指其控制的层数，一个模块的扇入数是指能直接控制该模块的模块数，一个模块的"扇出数"是指该模块直接控制的其他模块数。

55. 当一个模块直接使用另一个模块的内部数据，这种模块之间的耦合为内容耦合。

56. 在进行软件结构设计时应该遵循的最主要的原理是模块独立原理。

57. 变换型数据处理问题的工作过程大致分为三步，即取得数据，变换数据和给出数据。

58. 按数据流的类型,结构化设计方法有两种设计策略,它们是变换分析设计和事务分析设计。

59. 衡量模块的独立性的两个定性的度量标准是内聚度和耦合度。

60. 软件详细设计的主要任务是对算法和数据结构进行的详细设计，软件详细设计主要采用的方法是结构化程序设计。

61. 。

62. 过程描述语言可以用于算法和数据结构的描述。

63. 结构化程序设计方法是使用三种基本控制结构构造程序，程序的三种基本控制结构是顺序,选择和循环。

64. 在详细设计阶段,经常采用的工具有盒图、PAD 图、PDL 语言、判定表和判定树等工具，其中 PAD 图为自动分析数据提供了有力的工具.

65. 盒图也称为 N-S 图，种表达方式取消了流程线,它强迫程序员以结构化方式思考和解决问题.

66. 当模块中包含复杂的条件组合,只有判定表和判定树能够清晰地表达出各种动作之间的对应关系.

67. 模块的内部过程描述就是模块内部的算法设计，它的表达形式就是详细设计语言.

68. 程序的三种基本控制结构的共同特点是只有一个入口和一个出口。

69. 调试的目的是确定错误的位置和引起错误的原因,并加以改正。

70. 问题分析图也称为 PAD 图，这种图能更清晰地表达程序的逻辑结构。

71. 汇编语言是面向机器的，可以完成高级语言无法完成的特殊功能，如与外部设备之间的一些接口工作。

72. 注释是提高程序可读性的有效手段，好的程序注释占到程序总量的三分之一。

73. 回归测试：是指修改了旧代码后，重新进行测试以确认修改没有引入新的错误或导致其他代码产生错误。

74. 以详细设计说明书为输入,将该输入用某种程序设计语言翻译成计算机可以理解并最终可运行的代码的过程叫编码过程。

75. 面向对象的开发方法中,UML 是面向对象技术领域内占主导地位的标准建模语言。

76. 软件测试是执行程序并发现程序中潜伏的错误的过程。

77. 是对软件规格说明,软件设计和编码的最全面也是最后的审查。

78. 软件测试并不能发现软件中所有潜伏的错误，通过软件测试没有发现错误，不能说明软件是正确的。

79. 动态测试方法中根据测试用例的设计方法不同，分为黑盒和白盒两类.

80. 黑盒测试无需考虑模块内部的执行过程和程序结构，只要了解模块的功能即可。

81. 自顶向下的渐增式测试初期一般不可以并行进行。

82. 在现实项目中，路径测试和穷举测试是经常难以实现。

83. 单元测试的测试对象是程序模块。

84. 计算机辅助静态分析是软件测试方法中的静态测试方法之一。

85. 基本路径测试、循环覆盖测试、逻辑覆盖测试属于白盒测试技术。

86. 等价类划分、边界值分析测试等属于黑盒测试技术。

87. 逻辑覆盖一般包括语句覆盖、判定覆盖、 条件覆盖、条件/判定覆盖、边覆盖、路径覆盖等。

88. 将软件组装成系统的一种测试技术叫集成测试。

89. 软件测试中根据测试用例设计的方法的不同可分为黑盒测试和白盒测试两种，它们都属于动态测试。

90. 在设计测试用例时，边界值分析是用的最多的一种黑盒测试方法。

91. 在进行软件测试时，首先应当进行单元测试，然后再进行集成测试，最后再进行有效性测试。

92. 质量保证是为了保证产品和服务充分满足消费者要求的质量而进行的有计划．有组织的活动，质量保证是为了使用产品实现用户要求的功能。

93. 在结构测试用例设计中，有语句覆盖，条件覆盖，判定覆盖（即分支覆盖），路径覆盖，其中路径覆盖是最强的覆盖准则．

94. 自顶向下结合的渐增式测试法，在组合模块时有两种组合策略：深度优先策略和宽度优先策略。

95. 为了提高测试的效率，应该选择发现错误可能性大的数据作为测试数据。

96. 使用白盒测试方法时，确定测试数据应根据程序的内部逻辑和指定的覆盖标准。

97. 黑盒测试在设计测试用例时，主要需要研究需求规格说明与概要设计说明。

98. 软件按照设计的要求，在规定时间和条件下达到不出故障，持续运行的要求的质量特性称为可靠性。

99. 软件维护是软件生命周期的最后一个阶段，软件生命周期中所花费用最多的阶段是软件维护。

100. 在软件维护的内容中，有四种维护：纠错性维护，完善性维护，适应性维护，预防性维护，其中占维护活动工作量比例最高的是完善性维护，最少的一般是预防性维护。

101. 为改正软件系统中潜藏的错误而进行的维护活动称为改正性或纠错性维护。

102. 根据用户在软件使用过程中提出的建设性意见而进行的维护活动称为完善性维护，完善性维护是提高或完善软件的性能。

103. 为适应软件运行环境的变化而修改软件的活动称为适应性维护。

104. 为了进一步改善软件系统的可维护性和可靠性,并为以后的改进奠定基础的软件维护称为预防性维护。

105. 软件中因修改软件而造成的错误称为维护的副作用。

106. 软件生命周期终止的最典型原因是可维护性过差。

107. 非结构化维护用于软件的配置中只有源代码维护。

108. 维护中,因误删除一个标识符而引起的错误是编码副作用。

109. 软件可维护性,是指软件产品交付使用后,在实现改正潜伏的错误,改进性能等属性,适应环境变化等方面工作的难易程度。

110. 软件可靠性是指在给定的时间间隔内,程序成功运行的概率。

111. 软件可用性是指在给定的时间点,程序成功运行的概率。

112. 由于维护或在维护过程中其他一些不期望的行为引入的错误称为维护的 副作用。

113. 软件可修改性,是指允许对软件系统进行修改而不增加其复杂性。

114. 所有软件维护申请报告要按规定方式提出，该报告也称软件问题报告。

115. 边界类：用于描述目标软件系统与外部环境间的交互。

116. 实体类：表示目标软件系统中具有持久意义的信息项及其操作。

117. 控制类：完成用例任务的责任承担者，协调、控制其它类共同完成用例规定的功能或行为。

118. 软件危机：是指在计算机软件的开发和维护过程中所遇到的一系列严重问题。

119. 耦合性：也称块间联系，指软件系统结构中各模块间相互联系紧密程度的一种度量。

120. 软件生存周期模型：是描述软件开发过程中各种活动如何执行的模型.

121. 数据字典(DD)：是用来定义数据流图中的各个成分的具体含义的.它以一种准确的,无二义性的说明方式为系统的分析,设计及维护提供了有关元素的一致的定义和详细的描述.

122. 内聚性:是模块独立性的衡量标准之一,它是模块的功能强度的度量,即一个模块内部各个元素彼此结合的紧密程度的度量.

123. 黑盒测试：又称功能测试，是已经知道产品应该具有的功能，检验每个功能是否都能正常使用的测试方法。

124. 白盒测试：又称结构测试，是已经知道产品内部工作过程，检验产品内部动作是否按规定正常使用的测试方法。

125. 软件生存周期:就是从提出软件产品开始,直到该软件产品被淘汰的全过程,具体包括问题定义、可行性研究、需求分析、概要设计、详细设计、编码与测试、运行和维护。

126. 模块化：就是将程序划分为若干个独立模块的这样一个过程，其中每个模块完成一个特定子功能，每个模块既是相对独立的，又是相互联系的，它们共同完成系统指定的各项功能。

127. 桩模块：用于代替所测模块调用的子模块，桩模块可以做少量的数据操作。

128. 驱动模块：用于模拟被测模块的上级模块。它接收测试数据，把这些数据传送给所测模块，最后再输出实际测试结果。

129. 模块：模块是一个拥有明确定义的输入、输出和特性的程序实体。

130. 模块独立性：概括了把软件划分为模块时要遵守的准则，也是判断模块构造是否合理的标准，同时也是模块化、抽象及信息隐藏概念的直接产物。

131. 软件工程：即运用工程学的基本原理和方法来组织和管理软件生产。

132. 程序的可移植性:指把一个软件(或程序)从一台计算机环境移植到另一台计算机环境的容易程度。

133. 模块的作用范围:一个模块的作用范围是指受该模块内一个判定影响的所有模块的集合。

134. 信息隐藏：信息隐蔽是指在设计和确定模块时，使得一个模块内包含的信息(过程或数据)，对于不需要这些信息的其他模块来说是不能访问的。

135. 集成测试也称组装测试或联合测试。是指在单元测试的基础上，将所有模块按照设计要求组装成一个完整的系统进行的测试。组装模块的方式有两种：渐增式测试

和非渐增式测试。

136. 类：某些对象共同特征（属性和操作）的表示。

137. 对象：是现实世界中个体或事物的抽象表示，是其属性和相关操作的封装。

138. 继承：是现实世界中遗传关系的直接模型，它表示类间的内在联系以及对属性和操作的共享。泛化关系就是通常所说的继承关系。多重继承指的是一个子类可以同时多次继承同一个上层基类。

139. 聚集：现实世界中部分-整体关系的模拟。

140. 消息：对象与外部世界相互关联的唯一途径。

141. 螺旋模型的适应场合:支持需求不明确,特别是大型软件系统的开发,并支持面向规格说明,面向过程,面向对象等多种软件开发方法,是一种具有广阔前景的模型。

142. 软件开发风险分析实际上就是贯穿于软件工程过程中的一系列风险管理步骤,它包括以下内容：

1）风险标识；

2）风险估算；

3）风险评价；

4）风险驾驭和监控。

143. 需求分析的任务是确定待开发的软件系统"做什么".具体任务包括确定软件系统的功能需求,性能需求和运行环境约束,编制软件需求规格说明书,软件系统的验收测试准则和初步的用户手册。

144. 快速原型技术的基本思想是：在软件开发的早期,快速开发一个目标软件系统的原型,让用户对其进行评价并提出修改意见,然后开发人员根据用户的意见对原型进行改进。

145. 原型法模型一般适应的场合:它适合于那些不能预先确切定义需求的软件系统的开发,更适合于那些项目组成员(包括分析员,设计员,程序员和用户)不能很好交流或通信有困难的情况。

146. 软件复杂性度量的主要参数极其含义:1)规模:总共的指令数或源程序行数；2）难度:通常由程序中出现的操作数的数目所决定的量来表示；3）结构:通常用与程序结构有关的度量来表示；4）智能度:即算法的难易程序。

147. 非渐增式测试与渐增式测试的区别：

非渐增式测试：分别测试每个模块，再放在一起结合成所要的程序；

渐增式测试：将下一个要测试的模块同已测试好的模块放在一起测试，类推结合成所要的程序；

优缺点：

- 渐增式测试可以较早发现模块间的接口错误
- 非渐增式测试最后才组装，因此错误发现得晚。
- 非渐增式测试中发现错误后难以诊断定位
- 渐增式测试中，出现的错误往往跟最新加入的模块有关。
- 渐增式测试在不断集成的过程中使模块不断在新的条件下受到新的检测，测试更彻底。
- 渐增式测试较非渐增式测试费时。
- 非渐增式测试可以同时并行测试所有模块，能充分利用人力。

148. 软件工程目标:是研制、开发与生产出具有良好软件质量和费用合算的产品。软件工程的内容是：1）采用工程化方法和途径来开发与维护软件；2）应该开发和使用更好的软件工具；3）采取必要的管理措施。

149. 模块的内聚性包括的一般类型：

(1)偶然内聚

(2)逻辑内聚

(3)时间内聚

(4)通信内聚

(5)顺序内聚

(6)功能内聚

150. 软件测试的几个主要步骤：

1）模块测试

2）子系统测试

3）系统测试

4）验收测试

5）平行运行

151. 软件测试和调试的目的的区别：测试的目的是判断和发现软件是否有错误，调试的目的是定位软件错误并纠正错误。

152. 软件的可行性研究的目的：就是用最小的代价在尽可能短的时间内确定该软件项目是否能够开发，是否值得去开发。其实质是要进行一次简化、压缩了的需求分析、设计过程，要在较高层次上以较抽象的方式进行需求分析和设计过程。

153. 文档在软件工程中的作用：

(1)提高软件开发过程的能见度；

(2)提高开发效率；

(3)作为开发人员阶段工作成果和结束标志；

(4)记录开发过程的有关信息便于使用与维护；

(5)提供软件运行、维护和培训有关资料；

(6)便于用户了解软件功能、性能。

154. 按软件的功能进行划分，软件可以划分为系统软件、支撑软件和应用软件。

155. 软件生存周期一般可分为问题定义、可行性研究、需求分析、设计（概要设计与详细设计）、编码、测试、运行与维护阶段。

156. 可行性研究主要集中在以下四个方面：经济可行性、技术可行性、法律可行性和抉择。

157. 常见的软件概要设计方法有 3 大类：以数据流图为基础构造模块结构的结构化设计方法，以数据结构为基础构造模块的 Jackson 方法，以对象、类、继承和通信为基础的面向对象设计方法。

158. 单元测试一般以白盒测试为主，黑盒测试为辅。

159. 成本估计方法主要有自底向上估计、自顶向下估计和算法模型估计三种类型。

160. IPO 图是输入、处理和输出图的简称，它是美国 IBM 公司发展完善起来的一种图形工具。

161. 数据流图中的箭头表示数据流，椭圆或圆形表示数据处理，矩形表示数据的源点/终点。

162. 在结构化设计中，HIPO 图应用在总体设计阶段，由 IPO 图和层次图两部分组成。完整的软件结构通常用 HIPO 图来表示。

163. 复杂问题的对象模型通常由下述五个层次组成：主题层、类与对象层、结构层、属性层和服务层。

164. 实施精化设计的目标是基于模块的“高内聚低耦合”的原则，提高模块的独立性。

165. 面向对象的数据存储管理模式分为文件、关系数据库和面向对象数据库三种。

166. 能力成熟度模型分为 5 个等级：初始级、可重复级、已定义级、已管理级和优化级。

167. 向滞后的项目中增加人手会使得项目更加滞后。

168. 过程描述语言不能用于描述软件的系统结构。

169. 面向对象设计准则也要遵循弱耦合的原则，但是继承耦合则应该提高，紧密地继承耦合与高度的一般-特殊内聚是一致的。

170. 等价类划分方法将所有可能的输入数据划分成若干部分，然后从每一部分中选取少数有代表性的数据作为测试用例。

171. 模块化，信息隐藏，抽象和逐步求精的软件设计原则有助于得到高内聚，低耦合度的软件产品。

172. 在结构化分析方法中，数据流图是表达系统内部数据运动的图形化技术。

173. 在白盒测试技术测试用例的设计中，覆盖由弱到强依次是语句覆盖、判定覆盖、条件覆盖、判定/条件覆盖、条件组合覆盖、路径覆盖。

174. 一个模块的散出太大一般是因为缺乏中间层次，应当适当增加中间层次的控制模块。

175. 软件详细设计的主要任务是确定每个模块的算法和使用的数据结构。

176. 为了提高模块的独立性，模块内部最好是功能内聚。

177. 软件维护时，对测试阶段未发现的错误进行测试、诊断、定位、纠错，直至修改的回归测试过程称为改正性维护。适应性维护是随着软硬件环境变化而修改软件的过程。

178. 封装是把对象的属性和操作结合在一起，构成一个独立的对象，其内部信息对外界是隐蔽的，外界只能通过有限的接口与对象发生联系。

179. 面向对象的开发方法中，UML 语言是面向对象技术领域内占主导地位的标准建模语言。

180. 多态性意味着一个操作在不同的类中可以有不同的实现方式。

181. 单元测试的测试用例主要根据详细设计的结果来设计。

182. 可行性研究的目的是判断项目值得开发否。

183. 软件工程方法是在实践中不断发展着的方法，而早期的软件工程方法主要是指结构化方法，逐步求精法是结构化程序设计中的一种基本方法。

184. 计算机系统工程包含硬件、软件、人机及数据库工程。

185. 注释是提高程序可读性的有效手段，好的程序注释占到程序总量的 1/3。

186. 数据流图和数据字典共同构成系统的逻辑模型，没有数据字典数据流图就不严格，然而没有数据流图，数据字典也难以发挥作用。

187. 数据字典是开发数据库的第一步，而且是很有价值的一步。

188. 确认测试也称为验收测试，它的目标是验证软件的有效性。

189. 等价划分是一种黑盒测试技术，这种技术把程序的输入域划分成若干个数据类据此导出测试用例。

190. Rational 统一过程主要适用于大型的需求不断变化的复杂软件系统项目。

191. 程序设计风格的几个指导原则：

1） 嵌套的重数应加以限制；

2） 尽量少使用全局变量；
3） 使用有意义的变量名；
4） 把常见的局部优化工作留给编译程序去做；
5） 程序的格式应有助于读者理解程序。

二、练习

1、某些软件工程师不同意“目前国外许多软件开发组织，把60%以上的人力用于维护已有软件”的说法，他们争论说：“我并没有花费我的60%的时间去改正我所开发的程序中的错误。”请问你对上述争论有何看法？

答： 首先软件维护并非仅仅是改正程序中的错误，它还包括为了使软件适应变化了的环境而修改软件的活动，以及为了满足用户在使用软件的过程中提出的扩充或完善软件的新需求而修改软件的活动，甚至包括为了提高软件未来的可维护性或可靠性，而主动修改软件的活动，实际上为了消除程序中潜藏的错误而进行的改正性维护，仅占全部维护活动的1/5左右。

其次，目前国外许多软件开发组织，把60%以上的人力用于维护已有的软件，指的是软件开发组织内人力分配的整体状况，至于具体到软件组织内的每位软件工程师，则分工各不相同，有些人专职负责软件维护工作，他们的全部工作时间都花费在维护已有软件产品的工作上，另一些人专职负责软件开发工作，他们并不花费时间去维护已有的软件产品，还有一些人可能既要从事软件开发工作，又要监管软件维护工作。

最后软件维护人员并非只负责维护自己开发的程序，通常一名维护人员参与多个软件产品的维护工作。

2、假设你的任务是对一个已有的软件做重大修改，而且只允许你从下述文档中选取两份：（1）程序的规格说明；（2）程序的详细设计结果（自然语言描述加上某种设计工具表示）；（3）原程序清单（其中有适当数量的注解）。你将选取哪两份文档？为什么这样选取？

答：通常对一个已有的软件做重大修改，意味着对软件功能做较大变更或增加较多新功能，这往往需要修改软件的体系结构，因此了解原有软件的总体情况是很重要的，程序的规格说明书，准确的描述了对软件系统的数据要求、功能需求、性能需求、可靠性和可用性要求、出错处理需求、接口需求、约束、逆向需求及将来可能提出的需求，对了解已有软件的总体情况有很大帮助。在对已有软件做重大修改之前仔细阅读，认真研究这份文档，可以避免许多修改错误，因此应该选取这份文档。

有经验的软件工程师通过阅读含有适当数量注解的源程序，不难搞懂程序的实现算法，没有描述详细设计结果的文档并不会给维护工作带来太大困难，此外为了修改程序代码，原有程序的清单是必不可少的，因此为了对这个软件做重大修改，应该选取的第二份文档是源程序清单。

3、当一个十几年前开发出的程序，还在为其用户完成关键的业务工作时，是否有必要对它进行再工程，如果对它进行在工程经济上是否划算？

答：既然这个老程序目前还在为其用户完成关键的业务工作，可见他的业务重要程度相当高。但是在十几年前，软件工程还不像现在这样深入人心，软件过程管理还不成熟，那时开发出的软件，往往可维护性较差。一般来说用户的业务工作将持续相当长时间，也就是说这个程序很可能还要继续服役若干年，在这么长的时间里，它肯定还会经历若干次修改，与其每次花费很多人力物力来维护这个老程序,还不如在现代软件工程方法学指导下对它进行再工程。

粗看起来，在这个老程序还能正常完成用户的业务工作时重新开发它，在经济上似乎很不划算，其实不然。理由如下：

（1） 老程序的可维护性差，每次维护它将花费很高的代价。

（2） 在现代软件工程指导下开发出的软件的可维护性较好。此外，在再工程过程中可以建立起完整准确的文档，这进一步提高了软件的可维护性。因此，再工程之后每次维护它的代价将比在工程之前低得多，也就是说，可以节省很多人力物力。

（3） 正在工作的老程序，相当于一个原型，软件工程师从它那里可学到很多知识，从而大大提高了开发效率。

（4） 用户拥有长期使用该程序的丰富经验，容易提出对他的改进意见。通过再工程过程实现了用户的这些新需求之后，往往在相当长的一段时间内，不需要再维护。

（5） 现有的软件再工程工具，可以自动完成一部分再工程工作。

4、为什么说夏利牌汽车是小汽车类的特化，而发动机不是小汽车类的特化？

答：夏利牌汽车具有小汽车的全部属性和行为，它只不过是一种特定品牌的小汽车，因此夏利牌汽车可以从基类小汽车派生出来，也就是说夏利牌汽车是小汽车类的特化。发动机是组成小汽车的一种零件，小汽车还有车身，车灯，轮子等许多种其他零件，小汽车所具有的许多属性和行为发动机都不具有，因此发动机不能从小汽车类派生出来，它不是小汽车类的特化。

5、对象和属性之间有何区别？

答：对象是对客观世界实体的抽象，它是描述实体静态属性的数据和代表实体动态行为的操作结合在一起所构成的统一体。属性只不过是对象的一种特性，它是组成对象的一种成分。

6、什么是对象？它与传统的数据有何异同？

答：对象是用面向对象方法学开发软件时，对客观世界实体的抽象，它是由描述实体属性的数据及可以对这些数据施加的所有操作封装在一起构成的统一体。传统的数据是用传统方法学开发软件时，对客观世界实体的抽象，但是这种抽象是不全面的：数据只能描述实体的静态属性，不能描述实体的动态行为。必须从外界对数据施加操作才能改变数据，实现实体应有的行为。

对象与传统数据有本质的区别。它不是被动地等待外界对他施加操作，相反，他是进行处理的主体。必须发消息请求对象主动的执行他的某些操作，处理它的私有数据，而不能直接从外界对它的私有数据进行操作。

7、应该依据什么准则来评价用例图？

答：用例图是从用户的观点来描述系统的功能，因此，必须包含用户关心的所有关键功能。

8、应该依据什么准则来评价脚本？

答：脚本必须从用户的观点来描述每个重要的功能序列，因此，脚本应该能够说明系统的一类重要功能或具体的使用方法。

9、应该依据什么准则来评价状态图？

答：状态图应该描绘所有可能的状态转换。图中每条弧都要有一个引起状态转换的事件。从开始结点（初态）到每个结点（中间状态）以及从每个结点到最终结点（终态）都必须有一条路径。

10、用非正式分析法分析，确定下述杂货店问题中的对象。

一家杂货店想使其库存管理自动化。这家杂货店拥有能够记录顾客购买的所有商品的名称和数量的销售终端。顾客服务台也有类似的系统，以处理顾客的退货。它在码头有另一个终端处理供应商发货。肉食部和农产品部有终端用于输入由于损耗导致的损失和折扣。

答：非正式分析也称为词法分析，这种方法把用自然语言书写的需求描述中的名词作为候选的对象，从对杂货店问题的描述中可以找到下列名词作为对象的候选者：杂货店，库存，顾客，商品，名称，数量，销售终端，服务台终端，退货，码头，供应商，发货，肉食部，农产品部，损耗，损失，折扣。

其中“退货”初看起来像是动词，好像应该作为操作的候选者，但是经仔细分析可知“退货”包含货物名称，数量，价格等属性，实际上是一类对象。类似的发货也应该作为一类对象。

词法分析仅仅帮助我们找到一些候选的对象，接下来应该严格考察每个候选对象，从中删除不正确的或不必要的，只保留确实应该记录其信息或需要其提供服务的那些对象。

具体说到杂货店问题，名称和数量，实际上是顾客所购买商品的属性，不是独立存在的对象，销售终端和终端是同样的硬件设备，使用统一的名字终端就可以了。服务台和码头是放置某些终端的地点，他们与本软件关系不大，应该删掉。肉食部和农产品部是杂货店的两个部门，本软件并不处理杂货店的组织管理问题，因此他们不是本问题狱中的对象。但是从这两个部门可以想到杂货店有肉食品和农产品这样两类特殊的商品，应该把这两类商品作为问题域中的两类对象，损耗是导致损失和折扣的原因，不是独立的对象，综上所述杂货店问题域中的对象有：

杂货店，库存，顾客，商品，终端，退货，供应商，发货，肉食品，农产品，损失，折扣。

11、在面向对象设计过程中，为什么会调整对目标系统的需求，怎样调整需求？

答：有两种情况会导致修改由面向对象分析确定下来的系统需求，一是客户需求或系统外部环境发生了变化，二是分析员对问题与理解不透彻或缺乏领域专家帮助，以至面向对象分析模型，不能完整准确的反映客户的真实需求。

为了调整对目标系统的需求，通常只需简单的修改面向对象分析的结果，例如：添加或删掉一些类，从已有类派生出新类，调整某些类之间的关系等，然后把这些修改反映到问题域子系统中。

12、为了设计人机交互子系统为什么需要分类用户？

答：人机界面是提供给用户使用的，用户对人机界面的评价，在很大程度上由人的主观因素决定。用户的技能水平或职务不同，喜好和习惯也往往不同。因此，为了设计出符合用户需要的人机界面，应该了解和研究用户，根据用户类型设计出被他们所喜爱的用户界面。

13、问题空间和解空间有何区别？

答：问题空间是现实世界的一部分，它由现实世界中的实体组成。解空间实际上就是软件系统，它由实现解决方案的软件实体组成。

14、从面向对象分析阶段到面向对象设计阶段，对象模型有何变化？

答：在面向对象分析阶段建立的对象模型中，对象是对问题空间中实体的抽象，随着软

件开发过程进入面向对象设计阶段，这些对象逐渐变成了解空间中的实体。

15、为什么说分阶段的生命周期模型有助于软件项目管理？

答：软件是计算机系统的逻辑部件而不是物理部件，其固有的特点是缺乏可见性，因此管理和控制软件开发过程相当困难。分阶段的生命周期模型提高了软件项目的可见性，管理者可以把各个阶段任务的完成作为里程碑，来对软件开发过程进行管理。把阶段划分得更细，就能够更密切的监控软件项目的进展情况。

16、什么是里程碑？它应该有哪些特征？

答：里程碑是用来说明项目进展情况的事件，通常把一个开发活动的结束或一项开发任务的完成定义为一个里程碑。里程碑必须与软件开发工作的进展情况密切相关，而且，里程碑的完成必须非常明显，也就是说里程碑应该有很高的可见性。

17、假设要求你开发一个软件，该软件的功能是把读入的浮点数开平方，所得到的结果应该精确到小数点后 4 位，一旦实现并测试完之后，该产品将被抛弃。你打算选用哪种软件生命周期模型，请说明你做出选择的理由。

答：对这个软件的需求很明确，实现开平方功能的算法也很成熟，因此既无需通过原型来分析需求，也无需用原型来验证设计方案，此外，一旦实现并测试完之后，该产品将被抛弃，因此也无需使用有助于提高软件可维护性的增量模型或螺旋模型来开发该软件，综上所述，为了开发这个简单的软件，使用大多数人所熟悉的瀑布模型就可以了。

18、假设你被任命为一家软件公司的项目负责人，你的工作是：管理该公司已被广泛应用的字处理软件的新版本开发。由于市场竞争激烈，公司规定了严格的完成期限并且已对外公布，你打算采用哪种软件生命周期模型？为什么？

答：对这个项目的一个重要要求是：严格按照已对外公布了的日期完成产品开发工作。因此选择生命周期模型时，应该着重考虑哪种模型有助于加快产品开发的进度，使用增量模型开发软件时可以并行完成开发工作，因此能够加快开发进度。

这个项目是开发该公司已被广泛应用的字处理软件的新版本，从上述事实至少可以得出 3 点结论。

第一、旧版本相当于一个原型，通过收集用户对旧版本的反映，较容易确定对新版本的需求，没必要再专门建立一个原型系统来分析用户的需求；

第二，该公司的软件工程师对字处理软件很熟悉，有开发字处理软件的丰富经验，具有采用增量模型开发新版字处理软件所需要的技术水平；

第三，该软件受到广大用户的喜爱，今后很可能还要开发更新的版本，因此应该把该软

件的体系结构设计成开放式的，以利于今后的改进和扩充。

综上所述，采用增量模型来完成这个项目比较恰当。

19、在程序流程图中的每个节点，都必须有一条从开始节点到该节点本身的路径，以及一条从该节点到结束节点的路径。为什么数据流图没有关于节点之间可达性的类似规则？

答：数据流图不描述控制，因此在一个数据流图中，两个处理之间可能没有通路。如果几个处理都使用不同的输入数据，并生成不同的输出数据，并且一个处理的输出，不用做另一个处理的输入，那么他们之间就没有弧（边）。

20、设计下列伪码程序的分支覆盖和条件组合覆盖测试用例。

```
START
INPUT（A,B,C,D）
IF(A>0)AND(B>0)
    THEN X=A+B
    ELSE X=A-B
END
IF（C>A）OR（D<B）
THEN Y=C-D
    ELSE Y=C+D
END
PRINT(X,Y)
STOP.
```

答：（1）分支覆盖（即判定覆盖）标准为，不仅使每个语句至少执行一次，而且使每个判定表达式的每个分支都至少执行一次。

为做到分支覆盖，至少需要两组测试数据，以使每个判定表达式之值为真或为假各一次。下面是典型的测试用例：

①使两个判定表达式之值全为假

输入：A=-1，B=-2，C=-3，D=1

预期输出：X=1，Y=-2

②使两个判定表达式之值全为真

输入：A=1，B=2，C=3，D=1

预期输出：X=3，Y=2

（2）条件组合覆盖标准为：使得每个判定表达式中条件的各种可能组合都至少出现一次

本题程序中共有两个判定表达式，每个判定表达式中有两个简单条件，因此，总共有 8 种可能的条件组合，它们是：

① A>0,B>0

② A>0,B<=0

③ A<=0,B>0

④ A<=0,B<=0

⑤ C>A,D<B

⑥ C>A,D>=B

⑦ C<=A,D<B

⑧ C<=A,D>=B

下面的 4 个测试用例，可以使上面列出的 8 种条件组合每种至少出现一次：

- 实现①，⑤两种条件组合

  输入：A=1，B=1，C=2，D=0

  预期的输出：X=2，Y=2

- 实现②，⑥两种条件组合

  输入：A=1，B=0，C=2，D=1

  预期的输出：X=1，Y=1

- 实现③，⑦两种条件组合

  输入：A=0，B=1，C=-1，D=0

  预期的输出：X=-1，Y=-1

- 实现④，⑧两种条件组合

  输入：A=0，B=0，C=-1，D=1

  预期的输出：X=0，Y=0.

21、在测试一个长度为 24000 条指令的程序时，第一个月由甲乙两名测试员各自独立测试这个程序。经过一个月测试后，甲发现并改正 20 个错误，使 MTTF 达到 10h。与此同时，乙发现 24 个错误，其中 6 个甲也发现了，以后由甲一个人继续测试这个程序。

问：（1）刚开始测试时，程序中总共有多少个隐潜藏的错误？

（2）为使 MTTF 达到 60h，必须再改正多少个错误？

答：（1）本题中采用的是分别测试法，测试前程序中的总错误数：

$$\hat{B}_0 = \frac{B_2}{b_c} B_1 = 24 \div 6 \times 20 = 80。$$

（2）MTTF=24000/(K×($E_T - E_c(\tau)$))

由于：10=24000/（K×(80-20)）

得到：K=40

为使 MTTF 达到 60h，有：60=24000/（40×(80-$E_c(\tau)$)）

得：$E_c(\tau)$=70，即总共需要改正 70 个错误，由于测试员甲和乙分别测试时，测试员甲已经更正了 20 个错误，以后由甲一个人继续测试这个程序，因此还需要更正 70-20=50 个错误。

（提示：$MTTF = \frac{1}{K(E_T / I_T - E_c(\tau) / I_T)}$，

E_T：测试之前程序中错误总数；

I_T：程序长度（机器指令总数）；

τ：测试（包括调试）时间；

$E_c(\tau)$：在 0 至 τ 期间改正的错误数；

K：经验常数，在特定的情景下常需要自行确定。）