Week 9

AP Computer Science Principles

Homework bundle for this week

本周作业合集

exercises.lol

Contents 目录

AP Computer Science Principles Name: _____ Score: _____

1. A variable is best described as:

   ☐ a named box that stores one value at a time

   ☐ a permanent number that never changes

   ☐ a kind of error

   ☐ the program's output

2. Data abstraction helps a programmer by:

   ☐ hiding detail so complexity is easier to manage

   ☐ making programs run on more computers

   ☐ removing all the data

   ☐ adding more variables

3. What is $2 + 3 \times 4$?

   _____

4. A string is:

   ☐ an ordered sequence of characters

   ☐ a single number

   ☐ a true/false value

   ☐ a kind of loop

5. A Boolean expression always evaluates to:

   ☐ true or false

   ☐ any number

   ☐ a string

   ☐ a list

6. A conditional statement runs a block of code:

   ☐ only when its condition is true

   ☐ always, every time

   ☐ never

   ☐ in a random order

7. Repeating a block of code with a loop is called _____.

   _____

8. In score ← 10, the arrow means score _____ the value 10.

   _____

9. A _____ groups many values under one name, like all the marks of a class.

   _____

10. What is $(2 + 3) \times 4$?

    _____

3. Algorithms and Programming

Answers · 答案

AP Computer Science Principles

1. **a named box that stores one value at a time**

   The name is the label; the box holds the current value.

2. **hiding detail so complexity is easier to manage**

   Grouping data under one name hides needless detail.

3. **14**

   Multiplication first: $3 \times 4 = 12$, then $2 + 12 = 14$.

4. **an ordered sequence of characters**

   Order matters, so ”abc” $\neq$ ”cba”.

5. **true or false**

   Exactly one of two values: true or false.

6. **only when its condition is true**

   The condition decides whether the block runs.

7. **iteration**

   Iteration lets a short program do a lot of work.

8. **gets**

   Assignment stores the right-hand value into the left-hand name.

9. **list**

   A list is the most common abstract data type.

10. **20**

    Brackets first: $2 + 3 = 5$, then $5 \times 4 = 20$.

AP COMPUTER SCIENCE PRINCIPLES • EXERCISE SHEET

3.1 Variables and Assignments

Name: _____ Class: _____ Date: _____ **Total: 9 marks**

KEY WORDS variable 变量 • assignment 赋值 • data types 数据类型 • string 字符串

Objective

Build the skills to answer exam questions on **variables and assignments**.

You must be able to:

- explain how a **variable** 变量 is a named reference that stores a value
- use an **assignment** 赋值 statement to store or update a value
- describe how a variable can hold different **data types** 数据类型
- trace how a variable's value changes as a program runs

1 Worked examples

Study these first. Each one shows the method for a question type used later.

■ Variable and assignment

A **variable** is a named box that holds a value in memory. An **assignment** (setting `x` to `5`) stores or updates that value.

■ Data types

A variable can hold a **number**, a **string** (text), or a **Boolean** (true/false).

■ Tracing

Follow the value line by line: set `x` to `3`, then set `x` to `x + 2`, and `x` is now `5`.

2 Practice

2.1 Define a variable. [1]

2.2 A program sets `x` to `4`, then sets `x` to `x` × `3`. State the final value of `x`. [2]

2.3 State one benefit of using descriptive variable names. [1]

---

3 Exam-style questions

3.1 An assignment statement is used to [1]

- **A** print output
- **B** store a value in a variable
- **C** repeat code
- **D** define a function

3.2 Which of these values is a Boolean? [1]

- **A** `42`
- **B** `"hello"`
- **C** `true`
- **D** `3.14`

3.3 A program runs: set `a` to `10`; set `b` to `a + 5`; set `a` to `2`.

(a) State the final value of `a`. [1]

(b) State the final value of `b`. [1]

(c) State a good variable name for a value holding a student's age. [1]

Solutions

2.1 a named reference that stores a value in memory.

2.2 $4 \times 3 = 12$.

2.3 it makes the program easier to read and understand.

3.1 B.

3.2 C.

3.3 (a) `2`. (b) `15` (set before `a` changed). (c) any clear name such as `studentAge`.

AP COMPUTER SCIENCE PRINCIPLES • EXERCISE SHEET

3.2 Data Abstraction

Name: _____ Class: _____ Date: _____ **Total: 8 marks**

KEY WORDS abstraction 抽象 • list 列表 • data abstraction 数据抽象 • general 通用 • modeling 建模

Objective

Build the skills to answer exam questions on **data abstraction**.

You must be able to:

- explain how **data abstraction** 数据抽象 hides detail to manage complexity
- describe how a **list** 列表 groups many values under one name
- explain how using a list makes a program more **general** 通用
- relate data abstraction to real-world **modeling** 建模

1 Worked examples

Study these first. Each one shows the method for a question type used later.

■ Data abstraction

Data abstraction hides detail so the programmer can manage complexity. A **list** is an abstract data type that groups many values under **one name**.

■ More general

Storing scores as one list — `scores = [85, 90, 78]` — instead of three separate variables lets a single loop handle **any** number of values.

■ Modeling

Like real-world modeling, abstraction simplifies detail to focus on the essentials.

2 Practice

2.1 State what data abstraction does. [1]

2.2 State how a list makes a program more general. [1]

2.3 State one benefit of using a list instead of many separate variables. [1]

3 Exam-style questions

3.1 A list is an example of [1]

- **A** a data abstraction
- **B** a syntax error
- **C** a Boolean
- **D** a byte

3.2 Grouping many values under a single name is [1]

- **A** iteration
- **B** data abstraction
- **C** an event
- **D** compression

3.3 A program must store 100 test scores.

(a) State whether to use 100 separate variables or one list. [1]

(b) State one benefit of the choice in (a). [1]

(c) Name the general idea of hiding detail to manage complexity. [1]

Solutions

2.1 it hides detail to manage complexity in a program.

2.2 one list and one loop can handle any number of values.

2.3 it is easier to store, name, and process many values together.

3.1 A.

3.2 B.

3.3 (a) one list. (b) all scores can be processed with a single loop. (c) (data) abstraction.

2. Refer to your Personalized Project Reference when answering this question.

A. Consider the first iteration statement in the Procedure section of your Personalized Project Reference. Identify the variable(s) that determine when the iteration statement stops iterating. Identify specific values for the variable(s) that will cause the iteration statement to stop. Explain why these value(s) would cause the iteration statement to stop. If there are no such variable(s) or specific value(s), explain how the iteration statement stops.

B. Consider the procedure included in part (i) of the Procedure section of your Personalized Project Reference. Another programmer is planning to use your procedure in a different program and wants to write some test cases to check whether it will work. Write a procedure call with specific argument(s) that your procedure will accept but will cause your procedure to behave incorrectly. Describe the incorrect behavior. Explain why the incorrect behavior occurs as a result of this call. If it is not possible for a call with accepted arguments to cause your procedure to behave incorrectly, explain why this is the case for your procedure.

C. Consider the list included in part (i) of the List section of your Personalized Project Reference. Explain how the list uses abstraction to manage complexity in your program. Suppose the list was not included in your program. Describe how you would need to adjust the code segment in part (ii) of the List section of your Personalized Project Reference so that it has the same behavior. If it is not possible to have the same behavior without the use of the list, explain why.

STOP

END OF EXAM

2. Refer to your Personalized Project Reference when answering this question.

   (a) Consider the first conditional statement included in the Procedure section of your Personalized Project Reference. Describe your conditional statement, including its Boolean expression. Describe what the procedure does in general when the Boolean expression of this conditional statement evaluates to `false`.

   (b) Consider the procedure and procedure call identified in parts (i) and (ii) of the Procedure section of your Personalized Project Reference. Describe the outcome that your procedure call is intended to produce. Write a new procedure call with at least one different argument value that will produce the same outcome, if possible, and explain why this procedure call produces the same outcome. If it is not possible to write a new procedure call that produces the same outcome, explain why this is not possible.

   (c) Consider the procedure identified in part (i) of the Procedure section of your Personalized Project Reference. Identify the parameter(s) used in this procedure. Explain how your identified parameter(s) use abstraction to manage complexity in your program.

Write your responses to this question only on the designated pages in the separate Written Response booklet. If there are multiple parts to this question, write the part letter with your response.

GO ON TO THE NEXT PAGE.

STOP

END OF EXAM

AP COMPUTER SCIENCE PRINCIPLES • EXERCISE SHEET

3.3 Mathematical Expressions

Name: _____ Class: _____ Date: _____ **Total: 9 marks**

KEY WORDS remainder 余数 • modulus 取模 • order of operations 运算顺序 • arithmetic expression 算术表达式

Objective

Build the skills to answer exam questions on **mathematical expressions**.

You must be able to:

- evaluate an **arithmetic expression** 算术表达式
- apply the **order of operations** 运算顺序
- use the **modulus** 取模 operator (the remainder of integer division)
- reason about **integer** versus decimal results

1 Worked examples

Study these first. Each one shows the method for a question type used later.

■ Arithmetic and order of operations

Use $+$, $-$, $\times$, $\div$. Do $\times$ and $\div$ **before** $+$ and $-$; brackets first: $(2+3)\times 4 = 20$.

■ Modulus

`MOD` returns the **remainder** of integer division: $17 \bmod 5 = 2$ (since $17 = 3\times 5 + 2$).

■ Integer vs decimal

Combining whole numbers may give a whole or a decimal result — keep track of which.

2 Practice

2.1 Evaluate $2 + 3 \times 4$. [2]

2.2 Find 17 mod 5. [1]

2.3 State what the modulus operator returns. [1]

3 Exam-style questions

3.1 The modulus operator returns the [1]

- **A** quotient
- **B** remainder
- **C** sum
- **D** product

3.2 Using the order of operations, $10 - 2 \times 3$ equals [1]

- **A** 24
- **B** 4
- **C** 8
- **D** 6

3.3 Evaluate the following.

(a) $(6 + 4) \div 2$. [1]

(b) 20 mod 6. [1]

(c) State whether 20 mod 6 shows that 20 is a multiple of 6. [1]

Solutions

2.1 $\times$ first: $3 \times 4 = 12$, then $2 + 12 = 14$.

2.2 $17 \bmod 5 = 2$.

2.3 the remainder of an integer division.

3.1 B.

3.2 B — $2 \times 3 = 6$, then $10 - 6 = 4$.

3.3 (a) $10 \div 2 = 5$. (b) $20 \bmod 6 = 2$. (c) no — the remainder is 2, not 0, so 20 is not a multiple of 6.

AP COMPUTER SCIENCE PRINCIPLES • EXERCISE SHEET

3.4 Strings

Name: _____ Class: _____ Date: _____ **Total: 8 marks**

KEY WORDS string 字符串 • length 长度 • concatenation 拼接 • substring 子串 • assignment 赋值

Objective

Build the skills to answer exam questions on **strings**.

You must be able to:

- explain how a **string** 字符串 is an ordered sequence of characters
- use **concatenation** 拼接 to join strings
- find the **length** 长度 of a string
- extract a **substring** 子串 or a single character

1 Worked examples

Study these first. Each one shows the method for a question type used later.

■ Strings

A **string** is an ordered sequence of characters used to store text, e.g. `"hello"`.

■ Concatenation and length

Concatenation joins strings: `"foot" + "ball"` gives `"football"`. The **length** is the number of characters: `length("cat")` is `3`.

■ Substrings

You can extract a **substring** or an individual character from within a string, and use these operations to process **user input** such as names.

2 Practice

2.1 Define a string. [1]

2.2 State the result of concatenating `"sun"` and `"flower"`. [1]

2.3 State the length of the string `"data"`. [1]

3 Exam-style questions

3.1 Joining two strings into one is called [1]

- **A** iteration
- **B** concatenation
- **C** compression
- **D** assignment

3.2 The length of the string `"hello"` is [1]

- **A** 4
- **B** 5
- **C** 6
- **D** 1

3.3 A program greets a user by name.

(a) State the result of concatenating `"Hi, "` and `"Ana"`. [1]

(b) State the length of `"Ana"`. [1]

(c) State the data type used to store a name. [1]

Solutions

2.1 an ordered sequence of characters used to store text.

2.2 `"sunflower"`.

2.3 4.

3.1 B.

3.2 B.

3.3 (a) `"Hi, Ana"`. (b) 3. (c) a string.

AP COMPUTER SCIENCE PRINCIPLES • EXERCISE SHEET

3.5 Boolean Expressions

Name: _____ Class: _____ Date: _____ **Total: 9 marks**

KEY WORDS boolean expression 布尔表达式 • logical operators 逻辑运算符 • relational operators 关系运算符 • list 列表 • string 字符串

Objective

Build the skills to answer exam questions on **Boolean expressions**.

You must be able to:

- explain how a **Boolean expression** 布尔表达式 evaluates to true or false
- build one with **relational operators** 关系运算符 (equal to, greater than, less than)
- combine conditions with the **logical operators** 逻辑运算符 AND, OR, NOT
- predict the result of a compound Boolean expression

1 Worked examples

Study these first. Each one shows the method for a question type used later.

■ Boolean expression

Always evaluates to **true** or **false**.

■ Operators

- **Relational**: $=$, $>$, $<$ (and $\geq$, $\leq$).
- **Logical**: **AND** (both true), **OR** (either true), **NOT** (the opposite).

■ Compound expressions

Evaluate each part, then combine: $(5 > 3)$ AND $(2 > 4)$ = true AND false = **false**.

2 Practice

2.1 State what a Boolean expression evaluates to. [1]

2.2 Evaluate $(7 > 2)$ AND $(4 > 9)$. [2]

2.3 State what the NOT operator does. [1]

3 Exam-style questions

3.1 A Boolean expression evaluates to [1]

- **A** a number
- **B** a string
- **C** true or false
- **D** a list

3.2 The expression $(3 > 1)$ OR $(5 < 2)$ is [1]

- **A** true
- **B** false
- **C** undefined
- **D** 3

3.3 Given $a = 5$ and $b = 8$:

(a) evaluate $a < b$. [1]

(b) evaluate $(a < b)$ AND $(b < 10)$. [1]

(c) evaluate NOT $(a < b)$. [1]

Solutions

2.1 either true or false.

2.2 $7 > 2$ is true; $4 > 9$ is false; true AND false = **false**.

2.3 it reverses a Boolean value (true becomes false and vice versa).

3.1 C.

3.2 A — the first part is true, so OR is true.

3.3 (a) true. (b) true (both parts true). (c) false.

2. Refer to your Personalized Project Reference when answering this question.

A. Consider the first selection statement included in the Procedure section of your Personalized Project Reference. Identify the Boolean expression in this selection statement. Identify a specific value or set of values that will cause this expression to evaluate to `true`. Explain why the specified value(s) will cause the expression to evaluate to `true`.

B. Consider the procedure included in part (i) of the Procedure section of your Personalized Project Reference. Suppose another programmer modifies the code within this procedure. Describe a modification the other programmer could make that would cause this procedure to have a logic error. Describe how the behavior of this procedure would change because of the error.

C. Consider the list included in the List section of your Personalized Project Reference. Suppose another programmer adds several new elements to the end of the list. Explain how the code segment in part (ii) of the List section would need to be modified to account for the additional elements. If no changes to the code segment are necessary, explain why this is the case for your program.

STOP

END OF EXAM

2. Refer to your Personalized Project Reference when answering this question.

A. Consider the first selection statement included in the Procedure section of your Personalized Project Reference. Identify the Boolean expression in this selection statement. Identify a specific value or set of values that will cause this expression to evaluate to `false`. Explain why the specified value(s) will cause this expression to evaluate to `false`.

B. Consider the code segment in part (ii) of the List section of your Personalized Project Reference. Suppose another programmer modifies this code segment. Describe a modification the other programmer could make to this code segment that would result in a logic error. Explain why this modification would result in a logic error.

C. Consider the procedure identified in part (i) of the Procedure section of your Personalized Project Reference. Describe the functionality provided by this procedure. Explain how implementing this functionality as a procedure results in your program being easier to maintain than if the functionality were not implemented as a procedure.

STOP

END OF EXAM

2. Refer to your Personalized Project Reference when answering this question.

   (a) Consider the first conditional statement included in the Procedure section of your Personalized Project Reference. Describe your conditional statement, including its Boolean expression. Describe what the procedure does in general when the Boolean expression of this conditional statement evaluates to `false`.

   (b) Consider the procedure and procedure call identified in parts (i) and (ii) of the Procedure section of your Personalized Project Reference. Describe the outcome that your procedure call is intended to produce. Write a new procedure call with at least one different argument value that will produce the same outcome, if possible, and explain why this procedure call produces the same outcome. If it is not possible to write a new procedure call that produces the same outcome, explain why this is not possible.

   (c) Consider the procedure identified in part (i) of the Procedure section of your Personalized Project Reference. Identify the parameter(s) used in this procedure. Explain how your identified parameter(s) use abstraction to manage complexity in your program.

Write your responses to this question only on the designated pages in the separate Written Response booklet. If there are multiple parts to this question, write the part letter with your response.

GO ON TO THE NEXT PAGE.

STOP

END OF EXAM

AP COMPUTER SCIENCE PRINCIPLES • EXERCISE SHEET

3.6 Conditionals

Name: _____ Class: _____ Date: _____ **Total: 8 marks**

KEY WORDS selection 选择 • conditional 条件语句

Objective

Build the skills to answer exam questions on **conditionals**.

You must be able to:

- explain how a **conditional** 条件语句 runs a block only when a condition is true
- use an **if** statement with an **else** clause to choose between two paths
- describe how **selection** 选择 lets a program make decisions
- trace which branch executes for given inputs

1 Worked examples

Study these first. Each one shows the method for a question type used later.

If and if-else

An **if** statement runs its block only when the condition is **true**. Adding an **else** gives the path taken when it is **false**.

Selection

Selection is how a program **makes decisions** based on data. To predict behaviour, trace which branch runs.

Example

`if age >= 18: print "adult" else: print "minor"` prints `"adult"` only when age is at least 18.

2 Practice

2.1 State what a conditional (if) statement does. [1]

2.2 A program runs `if score >= 50: print "pass" else: print "fail"`. State the output when `score` is 40. [1]

2.3 State the term for a program making a decision based on data. [1]

3 Exam-style questions

3.1 An if statement runs its block [1]

- **A** always
- **B** only when the condition is true
- **C** never
- **D** exactly twice

3.2 An else clause runs when the condition is [1]

- **A** true
- **B** false
- **C** zero
- **D** missing

3.3 A program runs `if temp > 30: print "hot" else: print "ok"`.

(a) State the output when `temp` is 35. [1]

(b) State the output when `temp` is 20. [1]

(c) Name the process of choosing between the two branches. [1]

Solutions

2.1 it runs a block of code only when its condition is true.

2.2 `fail`.

2.3 selection.

3.1 B.

3.2 B.

3.3 (a) `hot`. (b) `ok`. (c) selection.

2. Refer to your Personalized Project Reference when answering this question.

A. Consider the first selection statement included in the Procedure section of your Personalized Project Reference. Identify the Boolean expression in this selection statement. Identify a specific value or set of values that will cause this expression to evaluate to `true`. Explain why the specified value(s) will cause the expression to evaluate to `true`.

B. Consider the procedure included in part (i) of the Procedure section of your Personalized Project Reference. Suppose another programmer modifies the code within this procedure. Describe a modification the other programmer could make that would cause this procedure to have a logic error. Describe how the behavior of this procedure would change because of the error.

C. Consider the list included in the List section of your Personalized Project Reference. Suppose another programmer adds several new elements to the end of the list. Explain how the code segment in part (ii) of the List section would need to be modified to account for the additional elements. If no changes to the code segment are necessary, explain why this is the case for your program.

STOP

END OF EXAM

2. Refer to your Personalized Project Reference when answering this question.

A. Consider the first selection statement included in the Procedure section of your Personalized Project Reference. Identify the Boolean expression in this selection statement. Identify a specific value or set of values that will cause this expression to evaluate to `false`. Explain why the specified value(s) will cause this expression to evaluate to `false`.

B. Consider the code segment in part (ii) of the List section of your Personalized Project Reference. Suppose another programmer modifies this code segment. Describe a modification the other programmer could make to this code segment that would result in a logic error. Explain why this modification would result in a logic error.

C. Consider the procedure identified in part (i) of the Procedure section of your Personalized Project Reference. Describe the functionality provided by this procedure. Explain how implementing this functionality as a procedure results in your program being easier to maintain than if the functionality were not implemented as a procedure.

STOP

END OF EXAM

2. Refer to your Personalized Project Reference when answering this question.

   (a) Consider the first conditional statement included in the Procedure section of your Personalized Project Reference. Describe your conditional statement, including its Boolean expression. Describe what the procedure does in general when the Boolean expression of this conditional statement evaluates to `false`.

   (b) Consider the procedure and procedure call identified in parts (i) and (ii) of the Procedure section of your Personalized Project Reference. Describe the outcome that your procedure call is intended to produce. Write a new procedure call with at least one different argument value that will produce the same outcome, if possible, and explain why this procedure call produces the same outcome. If it is not possible to write a new procedure call that produces the same outcome, explain why this is not possible.

   (c) Consider the procedure identified in part (i) of the Procedure section of your Personalized Project Reference. Identify the parameter(s) used in this procedure. Explain how your identified parameter(s) use abstraction to manage complexity in your program.

Write your responses to this question only on the designated pages in the separate Written Response booklet. If there are multiple parts to this question, write the part letter with your response.

GO ON TO THE NEXT PAGE.

STOP

END OF EXAM

AP COMPUTER SCIENCE PRINCIPLES • EXERCISE SHEET

3.7 Nested Conditionals

Name: _____ Class: _____ Date: _____ **Total: 8 marks**

KEY WORDS nested conditional 嵌套条件 • string 字符串 • variable 变量

Objective

Build the skills to answer exam questions on **nested conditionals**.

You must be able to:

- explain how a **nested conditional** 嵌套条件 places one conditional inside another
- use nested conditionals to select among **three or more** outcomes
- describe how an **else-if** structure tests conditions in sequence
- trace the path of execution for given inputs

1 Worked examples

Study these first. Each one shows the method for a question type used later.

■ Nested conditionals

A **nested conditional** is a conditional **inside** another. An **else-if** structure tests conditions **in sequence**, taking the first one that is true.

■ Choosing among many outcomes

`if score >= 90: "A" else if score >= 80: "B" else: "C"` selects one of three grades.

■ Tracing

Follow the tests in order and stop at the first true branch.

2 Practice

2.1 State what a nested conditional is. [1]

2.2 State what an else-if structure does. [1]

2.3 For `if s >= 90: "A" else if s >= 80: "B" else: "C"`, state the grade when `s` is 85. [1]

3 Exam-style questions

3.1 A nested conditional is [1]

- **A** a loop
- **B** a conditional inside another conditional
- **C** a variable
- **D** a string

3.2 An else-if structure is used to select among [1]

- **A** one outcome
- **B** exactly two outcomes
- **C** three or more outcomes
- **D** no outcomes

3.3 A program runs: `if m >= 50: (if m >= 75: print "distinction" else: print "pass") else: print "fail"`.

(a) State the output when `m` is 80. [1]

(b) State the output when `m` is 60. [1]

(c) State the output when `m` is 40. [1]

Solutions

2.1 a conditional statement placed inside another conditional.

2.2 it tests conditions in sequence and runs the first true one.

2.3 `B` ($85 < 90$ but $85 \geq 80$).

3.1 B.

3.2 C.

3.3 (a) `distinction`. (b) `pass`. (c) `fail`.

AP COMPUTER SCIENCE PRINCIPLES • EXERCISE SHEET

3.8 Iteration

Name: _____ Class: _____ Date: _____ **Total: 9 marks**

KEY WORDS iteration 迭代 • infinite loop 无限循环 • variable 变量 • list 列表 • string 字符串

Objective

Build the skills to answer exam questions on **iteration**.

You must be able to:

- explain how **iteration** 迭代 repeats a block of code using a loop
- use a **repeat** loop to run a block a fixed number of times
- use a condition-controlled loop that continues **until** a condition is true
- identify and avoid an **infinite loop** 无限循环

1 Worked examples

Study these first. Each one shows the method for a question type used later.

■ Iteration

A **loop** repeats a block of code. A **repeat** loop runs a fixed number of times; an **until** loop runs until a Boolean condition becomes true.

■ Infinite loops

A loop that never meets its stopping condition is an **infinite loop** — it never ends, so avoid it.

■ Tracing

Follow the variables across iterations: start `total` at 0, repeat "add 2" three times, and `total` becomes 6.

2 Practice

2.1 State what iteration does. [1]

2.2 A loop starts `total` at 0 and adds 5 each time, repeating 4 times. State the final total. [2]

2.3 State what an infinite loop is. [1]

3 Exam-style questions

3.1 A loop that repeats a block a fixed number of times is a [1]

- **A** conditional
- **B** repeat loop
- **C** variable
- **D** list

3.2 A loop that never meets its stopping condition is [1]

- **A** efficient
- **B** an infinite loop
- **C** a filter
- **D** a substring

3.3 A counter starts at 1 and doubles each time, repeated 3 times ($1 \to 2 \to 4 \to 8$).

(a) State the final value. [1]

(b) State how many times the block ran. [1]

(c) Name the general idea of repeating code. [1]

Solutions

2.1 it repeats a block of code (using a loop).

2.2 $0 + 5 + 5 + 5 + 5 = 20$.

2.3 a loop that never meets its stopping condition, so it runs forever.

3.1 B.

3.2 B.

3.3 (a) 8. (b) 3 times. (c) iteration.

2. Refer to your Personalized Project Reference when answering this question.

A. Consider the first iteration statement in the Procedure section of your Personalized Project Reference. Identify the variable(s) that determine when the iteration statement stops iterating. Identify specific values for the variable(s) that will cause the iteration statement to stop. Explain why these value(s) would cause the iteration statement to stop. If there are no such variable(s) or specific value(s), explain how the iteration statement stops.

B. Consider the procedure included in part (i) of the Procedure section of your Personalized Project Reference. Another programmer is planning to use your procedure in a different program and wants to write some test cases to check whether it will work. Write a procedure call with specific argument(s) that your procedure will accept but will cause your procedure to behave incorrectly. Describe the incorrect behavior. Explain why the incorrect behavior occurs as a result of this call. If it is not possible for a call with accepted arguments to cause your procedure to behave incorrectly, explain why this is the case for your procedure.

C. Consider the list included in part (i) of the List section of your Personalized Project Reference. Explain how the list uses abstraction to manage complexity in your program. Suppose the list was not included in your program. Describe how you would need to adjust the code segment in part (ii) of the List section of your Personalized Project Reference so that it has the same behavior. If it is not possible to have the same behavior without the use of the list, explain why.

STOP

END OF EXAM

2. Refer to your Personalized Project Reference when answering this question.

   (a) Consider the first iteration statement included in the Procedure section of your Personalized Project Reference. Describe what is being accomplished by the code in the body of the iteration statement.

   (b) Consider the procedure identified in part (i) of the Procedure section of your Personalized Project Reference. Write two calls to your procedure that each cause a different code segment in the procedure to execute. Describe the expected behavior of each call. If it is not possible for two calls to your procedure to cause different code segments to execute, explain why this is the case for your procedure.

   (c) Suppose another programmer provides you with a procedure called `checkValidity(value)` that returns `true` if a value passed as an argument is considered valid by the other programmer and returns `false` otherwise. Using the list identified in the List section of your Personalized Project Reference, explain in detailed steps an algorithm that uses `checkValidity` to check whether all elements in your list are considered valid by the other programmer. Your explanation must be detailed enough for someone else to write the program code for the algorithm that uses `checkValidity`.

Write your responses to this question only on the designated pages in the separate Written Response booklet. If there are multiple parts to this question, write the part letter with your response.

GO ON TO THE NEXT PAGE.

STOP

END OF EXAM

AP COMPUTER SCIENCE PRINCIPLES • EXERCISE SHEET

3.9 Developing Algorithms

Name: _____ Class: _____ Date: _____ **Total: 9 marks**

KEY WORDS algorithm 算法 • pseudocode 伪代码 • sequencing 顺序 • efficient 高效 • list 列表

Objective

Build the skills to answer exam questions on **developing algorithms**.

You must be able to:

- explain how an **algorithm** 算法 is a finite, step-by-step set of instructions
- express an algorithm using **sequencing** 顺序, selection, and iteration
- represent an algorithm as **pseudocode** 伪代码, a flowchart, or natural language
- refine an algorithm to be correct, clear, and **efficient** 高效

1 Worked examples

Study these first. Each one shows the method for a question type used later.

■ What an algorithm is

A **finite**, step-by-step set of instructions to complete a task, built from three building blocks: **sequencing**, **selection**, and **iteration**.

■ Representations

The same algorithm can be written as **pseudocode**, drawn as a **flowchart**, or described in **natural language**.

■ Different algorithms, same result

Two different algorithms can solve the same problem correctly; we compare them by clarity and **efficiency**.

2 Practice

2.1 Define an algorithm. [1]

2.2 Name the three building blocks of an algorithm. [2]

2.3 Name two ways to represent an algorithm. [1]

3 Exam-style questions

3.1 An algorithm must be [1]

- **A** infinite
- **B** finite and step-by-step
- **C** random
- **D** written without any words

3.2 Which of these is **not** a way to represent an algorithm? [1]

- **A** pseudocode
- **B** a flowchart
- **C** natural language
- **D** an overflow

3.3 Two students write different programs that both correctly sort a list.

(a) State whether both can be valid algorithms. [1]

(b) Name one quality by which to compare them. [1]

(c) Name the three building blocks of algorithms. [1]

Solutions

2.1 a finite, step-by-step set of instructions to complete a task.

2.2 sequencing, selection, iteration.

2.3 any two of: pseudocode, a flowchart, natural language.

3.1 B.

3.2 D.

3.3 (a) yes. (b) efficiency (or clarity). (c) sequencing, selection, iteration.

2. Refer to your Personalized Project Reference when answering this question.

   (a) Consider the first iteration statement included in the Procedure section of your Personalized Project Reference. Describe what is being accomplished by the code in the body of the iteration statement.

   (b) Consider the procedure identified in part (i) of the Procedure section of your Personalized Project Reference. Write two calls to your procedure that each cause a different code segment in the procedure to execute. Describe the expected behavior of each call. If it is not possible for two calls to your procedure to cause different code segments to execute, explain why this is the case for your procedure.

   (c) Suppose another programmer provides you with a procedure called `checkValidity(value)` that returns `true` if a value passed as an argument is considered valid by the other programmer and returns `false` otherwise. Using the list identified in the List section of your Personalized Project Reference, explain in detailed steps an algorithm that uses `checkValidity` to check whether all elements in your list are considered valid by the other programmer. Your explanation must be detailed enough for someone else to write the program code for the algorithm that uses `checkValidity`.

Write your responses to this question only on the designated pages in the separate Written Response booklet. If there are multiple parts to this question, write the part letter with your response.

GO ON TO THE NEXT PAGE.

STOP

END OF EXAM

AP COMPUTER SCIENCE PRINCIPLES • EXERCISE SHEET

3.10 Lists

Name: _____ Class: _____ Date: _____ **Total: 8 marks**

KEY WORDS list 列表 • elements 元素 • index 索引 • append 追加 • string 字符串

Objective

Build the skills to answer exam questions on **lists**.

You must be able to:

- explain how a **list** 列表 stores an ordered collection of **elements** 元素
- access or modify an element using its **index** 索引 position
- use **iteration** to traverse a list
- apply list operations such as **append** 追加, insert, and remove

1 Worked examples

Study these first. Each one shows the method for a question type used later.

List and index

A **list** stores an ordered collection of values under one name. Each element has an **index** (position); here the first element is at index 1.

Operations

append (add to the end), **insert** (add at a position), **remove** (delete an element).

Traversing

Use **iteration** to visit and process every element. For `scores = [8, 5, 9]`, `scores[1]` is 8 and `scores[3]` is 9.

2 Practice

2.1 Define a list. [1]

2.2 For the list $[4, 7, 2, 9]$, state the element at index 2. [1]

2.3 Name one operation that changes a list's contents. [1]

3 Exam-style questions

3.1 A list stores [1]

- **A** one value only
- **B** an ordered collection of values
- **C** only text
- **D** a single Boolean

3.2 To process every element of a list, you use [1]

- **A** a filter only
- **B** iteration
- **C** a substring
- **D** compression

3.3 A list `names = ["Ana", "Ben", "Cara"]` (index 1 is first).

(a) State `names[1]`. [1]

(b) State the length of the list. [1]

(c) Name the operation that adds `"Dan"` to the end. [1]

Solutions

2.1 an ordered collection of values (elements) stored under one name.

2.2 7.

2.3 any one of: append, insert, remove.

3.1 B.

3.2 B.

3.3 (a) `"Ana"`. (b) 3. (c) append.

2. Refer to your Personalized Project Reference when answering this question.

A. Consider the first iteration statement in the Procedure section of your Personalized Project Reference. Identify the variable(s) that determine when the iteration statement stops iterating. Identify specific values for the variable(s) that will cause the iteration statement to stop. Explain why these value(s) would cause the iteration statement to stop. If there are no such variable(s) or specific value(s), explain how the iteration statement stops.

B. Consider the procedure included in part (i) of the Procedure section of your Personalized Project Reference. Another programmer is planning to use your procedure in a different program and wants to write some test cases to check whether it will work. Write a procedure call with specific argument(s) that your procedure will accept but will cause your procedure to behave incorrectly. Describe the incorrect behavior. Explain why the incorrect behavior occurs as a result of this call. If it is not possible for a call with accepted arguments to cause your procedure to behave incorrectly, explain why this is the case for your procedure.

C. Consider the list included in part (i) of the List section of your Personalized Project Reference. Explain how the list uses abstraction to manage complexity in your program. Suppose the list was not included in your program. Describe how you would need to adjust the code segment in part (ii) of the List section of your Personalized Project Reference so that it has the same behavior. If it is not possible to have the same behavior without the use of the list, explain why.

STOP

END OF EXAM

2. Refer to your Personalized Project Reference when answering this question.

A. Consider the first selection statement included in the Procedure section of your Personalized Project Reference. Identify the Boolean expression in this selection statement. Identify a specific value or set of values that will cause this expression to evaluate to `true`. Explain why the specified value(s) will cause the expression to evaluate to `true`.

B. Consider the procedure included in part (i) of the Procedure section of your Personalized Project Reference. Suppose another programmer modifies the code within this procedure. Describe a modification the other programmer could make that would cause this procedure to have a logic error. Describe how the behavior of this procedure would change because of the error.

C. Consider the list included in the List section of your Personalized Project Reference. Suppose another programmer adds several new elements to the end of the list. Explain how the code segment in part (ii) of the List section would need to be modified to account for the additional elements. If no changes to the code segment are necessary, explain why this is the case for your program.

STOP

END OF EXAM

2. Refer to your Personalized Project Reference when answering this question.

A. Consider the first selection statement included in the Procedure section of your Personalized Project Reference. Identify the Boolean expression in this selection statement. Identify a specific value or set of values that will cause this expression to evaluate to `false`. Explain why the specified value(s) will cause this expression to evaluate to `false`.

B. Consider the code segment in part (ii) of the List section of your Personalized Project Reference. Suppose another programmer modifies this code segment. Describe a modification the other programmer could make to this code segment that would result in a logic error. Explain why this modification would result in a logic error.

C. Consider the procedure identified in part (i) of the Procedure section of your Personalized Project Reference. Describe the functionality provided by this procedure. Explain how implementing this functionality as a procedure results in your program being easier to maintain than if the functionality were not implemented as a procedure.

STOP

END OF EXAM

2. Refer to your Personalized Project Reference when answering this question.

   (a) Consider the first iteration statement included in the Procedure section of your Personalized Project Reference. Describe what is being accomplished by the code in the body of the iteration statement.

   (b) Consider the procedure identified in part (i) of the Procedure section of your Personalized Project Reference. Write two calls to your procedure that each cause a different code segment in the procedure to execute. Describe the expected behavior of each call. If it is not possible for two calls to your procedure to cause different code segments to execute, explain why this is the case for your procedure.

   (c) Suppose another programmer provides you with a procedure called `checkValidity(value)` that returns `true` if a value passed as an argument is considered valid by the other programmer and returns `false` otherwise. Using the list identified in the List section of your Personalized Project Reference, explain in detailed steps an algorithm that uses `checkValidity` to check whether all elements in your list are considered valid by the other programmer. Your explanation must be detailed enough for someone else to write the program code for the algorithm that uses `checkValidity`.

Write your responses to this question only on the designated pages in the separate Written Response booklet. If there are multiple parts to this question, write the part letter with your response.

GO ON TO THE NEXT PAGE.

STOP

END OF EXAM

AP COMPUTER SCIENCE PRINCIPLES • EXERCISE SHEET

3.11 Binary Search

Name: _____ Class: _____ Date: _____ **Total: 8 marks**

KEY WORDS binary search 二分查找 • sorted 已排序 • halves 减半 • list 列表

Objective

Build the skills to answer exam questions on **binary search**.

You must be able to:

- explain how **binary search** 二分查找 locates a target in a **sorted** list
- describe how each comparison **halves** 减半 the remaining portion
- state the requirement that the data must be **sorted** 已排序
- compare binary search with a linear search

1 Worked examples

Study these first. Each one shows the method for a question type used later.

■ Binary search

On a **sorted** list, look at the **middle** element. If the target is smaller, search the **left** half; if larger, search the **right** half. Each comparison **halves** the items still to check.

■ Requirement

The list **must be sorted** first, or binary search fails.

■ Versus linear search

A linear search checks each element in turn; binary search is far faster on large sorted lists (about $\log_2 n$ steps instead of n).

2 Practice

2.1 State the requirement a list must meet for binary search to work. [1]

2.2 State what each comparison in a binary search does to the remaining items. [1]

2.3 State one advantage of binary search over linear search. [1]

3 Exam-style questions

3.1 Binary search requires the list to be [1]

- **A** sorted
- **B** empty
- **C** reversed
- **D** in random order

3.2 Each step of a binary search [1]

- **A** doubles
- **B** halves
- **C** ignores
- **D** shuffles

the remaining items to search.

3.3 A sorted list has 16 items.

(a) Name the search method that halves the list at each step. [1]

(b) State roughly how many steps it needs in the worst case ($16 \to 8 \to 4 \to 2 \to 1$). [1]

(c) State whether binary search works on an **unsorted** list. [1]

Solutions

2.1 it must be sorted.

2.2 it halves them (discards one half).

2.3 it is much faster on large sorted lists (fewer comparisons).

3.1 A.

3.2 B.

3.3 (a) binary search. (b) about 4 steps. (c) no.

AP COMPUTER SCIENCE PRINCIPLES • EXERCISE SHEET

3.12 Calling Procedures

Name: _____ Class: _____ Date: _____ **Total: 8 marks**

KEY WORDS procedure 过程 • arguments 实参 • return value 返回值 • abstraction 抽象 • list 列表

Objective

Build the skills to answer exam questions on **calling procedures**.

You must be able to:

- explain how a **procedure** 过程 (function or method) is a named block of reusable code
- **call** a procedure by name to run its instructions
- pass **arguments** 实参 through its parameters
- use a **return value** 返回值 in a later expression

1 Worked examples

Study these first. Each one shows the method for a question type used later.

■ Procedures

A **procedure** (also called a function or method) is a **named block of reusable code**. You **call** it by name to run its instructions from anywhere.

■ Arguments and return values

Values passed in are **arguments**, received through the procedure's **parameters**. A procedure may produce a **return value** that the caller uses.

■ Abstraction

Procedures support **abstraction** by hiding their internal detail from the caller: `area(5, 3)` returns 15 without the caller seeing how.

2 Practice

2.1 Define a procedure. [1]

2.2 State how you run a procedure from elsewhere in a program. [1]

2.3 State what a return value is. [1]

3 Exam-style questions

3.1 A procedure is [1]

- **A** a single variable
- **B** a named block of reusable code
- **C** a loop
- **D** a data type

3.2 Values passed into a procedure are called [1]

- **A** return values
- **B** arguments
- **C** comments
- **D** lists

3.3 A procedure `double(n)` returns `n` × `2`.

(a) State the value of `double(6)`. [1]

(b) Name what `6` is when it is passed in. [1]

(c) State how procedures support abstraction. [1]

Solutions

2.1 a named block of reusable code (a function or method).

2.2 call it by its name.

2.3 a value that a procedure produces and hands back to the caller.

3.1 B.

3.2 B.

3.3 (a) 12. (b) an argument. (c) they hide the implementation detail from the caller.

2. Refer to your Personalized Project Reference when answering this question.

A. Consider the first iteration statement in the Procedure section of your Personalized Project Reference. Identify the variable(s) that determine when the iteration statement stops iterating. Identify specific values for the variable(s) that will cause the iteration statement to stop. Explain why these value(s) would cause the iteration statement to stop. If there are no such variable(s) or specific value(s), explain how the iteration statement stops.

B. Consider the procedure included in part (i) of the Procedure section of your Personalized Project Reference. Another programmer is planning to use your procedure in a different program and wants to write some test cases to check whether it will work. Write a procedure call with specific argument(s) that your procedure will accept but will cause your procedure to behave incorrectly. Describe the incorrect behavior. Explain why the incorrect behavior occurs as a result of this call. If it is not possible for a call with accepted arguments to cause your procedure to behave incorrectly, explain why this is the case for your procedure.

C. Consider the list included in part (i) of the List section of your Personalized Project Reference. Explain how the list uses abstraction to manage complexity in your program. Suppose the list was not included in your program. Describe how you would need to adjust the code segment in part (ii) of the List section of your Personalized Project Reference so that it has the same behavior. If it is not possible to have the same behavior without the use of the list, explain why.

STOP

END OF EXAM

2. Refer to your Personalized Project Reference when answering this question.

   (a) Consider the first iteration statement included in the Procedure section of your Personalized Project Reference. Describe what is being accomplished by the code in the body of the iteration statement.

   (b) Consider the procedure identified in part (i) of the Procedure section of your Personalized Project Reference. Write two calls to your procedure that each cause a different code segment in the procedure to execute. Describe the expected behavior of each call. If it is not possible for two calls to your procedure to cause different code segments to execute, explain why this is the case for your procedure.

   (c) Suppose another programmer provides you with a procedure called `checkValidity(value)` that returns `true` if a value passed as an argument is considered valid by the other programmer and returns `false` otherwise. Using the list identified in the List section of your Personalized Project Reference, explain in detailed steps an algorithm that uses `checkValidity` to check whether all elements in your list are considered valid by the other programmer. Your explanation must be detailed enough for someone else to write the program code for the algorithm that uses `checkValidity`.

Write your responses to this question only on the designated pages in the separate Written Response booklet. If there are multiple parts to this question, write the part letter with your response.

GO ON TO THE NEXT PAGE.

STOP

END OF EXAM

2. Refer to your Personalized Project Reference when answering this question.

   (a) Consider the first conditional statement included in the Procedure section of your Personalized Project Reference. Describe your conditional statement, including its Boolean expression. Describe what the procedure does in general when the Boolean expression of this conditional statement evaluates to `false`.

   (b) Consider the procedure and procedure call identified in parts (i) and (ii) of the Procedure section of your Personalized Project Reference. Describe the outcome that your procedure call is intended to produce. Write a new procedure call with at least one different argument value that will produce the same outcome, if possible, and explain why this procedure call produces the same outcome. If it is not possible to write a new procedure call that produces the same outcome, explain why this is not possible.

   (c) Consider the procedure identified in part (i) of the Procedure section of your Personalized Project Reference. Identify the parameter(s) used in this procedure. Explain how your identified parameter(s) use abstraction to manage complexity in your program.

Write your responses to this question only on the designated pages in the separate Written Response booklet. If there are multiple parts to this question, write the part letter with your response.

GO ON TO THE NEXT PAGE.

STOP

END OF EXAM

AP COMPUTER SCIENCE PRINCIPLES • EXERCISE SHEET

3.13 Developing Procedures

Name: _____ Class: _____ Date: _____ **Total: 9 marks**

KEY WORDS parameters 参数 • duplication 重复 • procedural abstraction 过程抽象 • abstraction 抽象 • list 列表

Objective

Build the skills to answer exam questions on **developing procedures**.

You must be able to:

- **define** a procedure with a name, parameters, and body
- use **parameters** 参数 so a procedure works with different inputs
- apply **procedural abstraction** 过程抽象 to break a problem into parts
- explain how procedures reduce **duplication** 重复

1 Worked examples

Study these first. Each one shows the method for a question type used later.

■ Defining a procedure

Give it a **name**, a list of **parameters**, and a **body** of instructions.

■ Parameters

Parameters let one procedure work with **different input values** each time it is called.

■ Procedural abstraction

Breaking a large problem into smaller, well-named procedures reduces **duplication** and improves readability and reuse.

2 Practice

2.1 State the three parts of a procedure definition. [2]

2.2 State the purpose of a parameter. [1]

2.3 State one benefit of using procedures. [1]

3 Exam-style questions

3.1 Breaking a large problem into smaller procedures is [1]

- **A** iteration
- **B** procedural abstraction
- **C** compression
- **D** an event

3.2 A parameter lets a procedure [1]

- **A** run only once
- **B** work with different input values
- **C** never return anything
- **D** store metadata

3.3 A program repeats the same five lines in three different places.

(a) State how to avoid this duplication. [1]

(b) State one benefit of doing so. [1]

(c) Name the general idea involved. [1]

Solutions

2.1 a name, parameters, and a body.

2.2 to let the procedure work with different input values.

2.3 any one of: less duplication, easier reuse, better readability.

3.1 B.

3.2 B.

3.3 (a) put the lines in a procedure and call it in each place. (b) less duplication (or easier to maintain). (c) procedural abstraction.

2. Refer to your Personalized Project Reference when answering this question.

A. Consider the first selection statement included in the Procedure section of your Personalized Project Reference. Identify the Boolean expression in this selection statement. Identify a specific value or set of values that will cause this expression to evaluate to `true`. Explain why the specified value(s) will cause the expression to evaluate to `true`.

B. Consider the procedure included in part (i) of the Procedure section of your Personalized Project Reference. Suppose another programmer modifies the code within this procedure. Describe a modification the other programmer could make that would cause this procedure to have a logic error. Describe how the behavior of this procedure would change because of the error.

C. Consider the list included in the List section of your Personalized Project Reference. Suppose another programmer adds several new elements to the end of the list. Explain how the code segment in part (ii) of the List section would need to be modified to account for the additional elements. If no changes to the code segment are necessary, explain why this is the case for your program.

STOP

END OF EXAM

2. Refer to your Personalized Project Reference when answering this question.

A. Consider the first selection statement included in the Procedure section of your Personalized Project Reference. Identify the Boolean expression in this selection statement. Identify a specific value or set of values that will cause this expression to evaluate to `false`. Explain why the specified value(s) will cause this expression to evaluate to `false`.

B. Consider the code segment in part (ii) of the List section of your Personalized Project Reference. Suppose another programmer modifies this code segment. Describe a modification the other programmer could make to this code segment that would result in a logic error. Explain why this modification would result in a logic error.

C. Consider the procedure identified in part (i) of the Procedure section of your Personalized Project Reference. Describe the functionality provided by this procedure. Explain how implementing this functionality as a procedure results in your program being easier to maintain than if the functionality were not implemented as a procedure.

STOP

END OF EXAM

2. Refer to your Personalized Project Reference when answering this question.

   (a) Consider the first iteration statement included in the Procedure section of your Personalized Project Reference. Describe what is being accomplished by the code in the body of the iteration statement.

   (b) Consider the procedure identified in part (i) of the Procedure section of your Personalized Project Reference. Write two calls to your procedure that each cause a different code segment in the procedure to execute. Describe the expected behavior of each call. If it is not possible for two calls to your procedure to cause different code segments to execute, explain why this is the case for your procedure.

   (c) Suppose another programmer provides you with a procedure called `checkValidity(value)` that returns `true` if a value passed as an argument is considered valid by the other programmer and returns `false` otherwise. Using the list identified in the List section of your Personalized Project Reference, explain in detailed steps an algorithm that uses `checkValidity` to check whether all elements in your list are considered valid by the other programmer. Your explanation must be detailed enough for someone else to write the program code for the algorithm that uses `checkValidity`.

Write your responses to this question only on the designated pages in the separate Written Response booklet. If there are multiple parts to this question, write the part letter with your response.

GO ON TO THE NEXT PAGE.

STOP

END OF EXAM

2. Refer to your Personalized Project Reference when answering this question.

   (a) Consider the first conditional statement included in the Procedure section of your Personalized Project Reference. Describe your conditional statement, including its Boolean expression. Describe what the procedure does in general when the Boolean expression of this conditional statement evaluates to `false`.

   (b) Consider the procedure and procedure call identified in parts (i) and (ii) of the Procedure section of your Personalized Project Reference. Describe the outcome that your procedure call is intended to produce. Write a new procedure call with at least one different argument value that will produce the same outcome, if possible, and explain why this procedure call produces the same outcome. If it is not possible to write a new procedure call that produces the same outcome, explain why this is not possible.

   (c) Consider the procedure identified in part (i) of the Procedure section of your Personalized Project Reference. Identify the parameter(s) used in this procedure. Explain how your identified parameter(s) use abstraction to manage complexity in your program.

Write your responses to this question only on the designated pages in the separate Written Response booklet. If there are multiple parts to this question, write the part letter with your response.

GO ON TO THE NEXT PAGE.

STOP

END OF EXAM

AP COMPUTER SCIENCE PRINCIPLES • EXERCISE SHEET

3.14 Libraries

Name: _____ Class: _____ Date: _____ **Total: 8 marks**

KEY WORDS library 库 • api 应用程序接口 • functionality 功能 • abstraction 抽象 • list 列表

Objective

Build the skills to answer exam questions on **libraries**.

You must be able to:

- explain how a **library** 库 is a collection of reusable procedures
- describe how an **API** 应用程序接口 documents how to use a library
- call library procedures to add **functionality** 功能 without writing it from scratch
- select an appropriate library for a problem

1 Worked examples

Study these first. Each one shows the method for a question type used later.

■ Library and API

A **library** is a collection of procedures that can be reused across programs. An **API** (application program interface) documents **how to use** the library — what procedures exist and what they need.

■ Why use one

Calling library procedures adds **functionality without writing it from scratch**, and libraries support **abstraction** by hiding complex implementation detail.

2 Practice

2.1 Define a software library. [1]

2.2 State what an API documents. [1]

2.3 State one benefit of using a library. [1]

3 Exam-style questions

3.1 A library is a collection of [1]

- **A** variables
- **B** reusable procedures
- **C** errors
- **D** bytes

3.2 An API is used to [1]

- **A** store data
- **B** document how to use a library
- **C** compress files
- **D** sort a list

3.3 A developer needs graph-drawing features for a program.

(a) State whether to write them from scratch or use a library. [1]

(b) Name what documents the library's use. [1]

(c) State how libraries support abstraction. [1]

Solutions

2.1 a collection of procedures that can be reused across programs.

2.2 how to use the library (its available procedures and how to call them).

2.3 it adds functionality without writing it from scratch.

3.1 B.

3.2 B.

3.3 (a) use a library. (b) the API. (c) they hide the complex implementation detail.

AP COMPUTER SCIENCE PRINCIPLES • EXERCISE SHEET

3.15 Random Values

Name: _____ Class: _____ Date: _____ **Total: 8 marks**

KEY WORDS random 随机 • variability 变化 • unpredictable 不可预测 • string 字符串

Objective

Build the skills to answer exam questions on **random values**.

You must be able to:

- explain how a program generates a **random** 随机 value within a range
- use a random number to introduce **variability** 变化
- apply random values to model **unpredictable** 不可预测 events
- explain why repeated runs can produce different outputs

1 Worked examples

Study these first. Each one shows the method for a question type used later.

■ Random values

A program can generate a **random** value within a specified range to introduce **variability** into its behaviour.

■ Modelling chance

Random values model **unpredictable** events such as a dice roll or coin flip.

■ Different outputs

Because the value is random, running the same program again can produce a **different output** each time — `random(1, 6)` gives some whole number from 1 to 6.

2 Practice

2.1 State one use of random values in a program. [1]

2.2 State why a program using randomness can give different results each run. [1]

2.3 State one everyday event that can be modelled with random values. [1]

3 Exam-style questions

3.1 A program generates a random number to add [1]

- **A** a syntax error
- **B** variability
- **C** a comment
- **D** a byte

3.2 Rolling a die in a program is modelled with a [1]

- **A** fixed value
- **B** random value
- **C** string
- **D** loop

3.3 A dice game uses `random(1, 6)`.

(a) State the possible outputs. [1]

(b) State whether two runs must give the same result. [1]

(c) State one reason to use randomness. [1]

Solutions

2.1 any one of: model chance, add variability, shuffle, simulate an event.

2.2 the random value can differ each time the program runs.

2.3 a dice roll, a coin flip, or shuffling cards (any one).

3.1 B.

3.2 B.

3.3 (a) a whole number from 1 to 6. (b) no. (c) to model an unpredictable event (or add variety).

AP COMPUTER SCIENCE PRINCIPLES • EXERCISE SHEET

3.16 Simulations

Name: _____ Class: _____ Date: _____ **Total: 8 marks**

KEY WORDS simulation 模拟 • phenomenon 现象 • risk 风险 • limitations 局限性 • list 列表

Objective

Build the skills to answer exam questions on **simulations**.

You must be able to:

- explain how a **simulation** 模拟 models a real-world **phenomenon** 现象
- describe how simulations use **abstraction** to simplify reality
- identify benefits such as lower cost and reduced **risk** 风险
- recognize the **limitations** 局限性 of a simulation

1 Worked examples

Study these first. Each one shows the method for a question type used later.

■ What a simulation is

A **simulation** is a program that **models a real-world phenomenon**, using **abstraction** to simplify and represent reality.

■ Benefits

It can be **cheaper** and **safer** than a real trial, and **randomness** can make it more realistic.

■ Limitations

A simulation leaves out detail through its assumptions, so its results may not perfectly match reality.

2 Practice

2.1 Define a simulation. [1]

2.2 State one benefit of a simulation over a real trial. [1]

2.3 State one limitation of a simulation. [1]

3 Exam-style questions

3.1 A simulation is a program that [1]

- **A** stores data
- **B** models a real-world phenomenon
- **C** compresses files
- **D** sorts a list

3.2 One benefit of a simulation is [1]

- **A** more risk
- **B** lower cost and reduced risk
- **C** guaranteed perfect accuracy
- **D** having no assumptions

3.3 Engineers simulate a bridge design before building it.

(a) State one benefit of doing this. [1]

(b) State how randomness could make the simulation more realistic. [1]

(c) State one limitation of the simulation. [1]

Solutions

2.1 a program that models a real-world phenomenon.

2.2 it is cheaper or safer than a real trial.

2.3 it simplifies reality through its assumptions, so it may miss real details.

3.1 B.

3.2 B.

3.3 (a) lower cost or less risk than building and testing a real bridge. (b) it could model varied loads or weather conditions. (c) its simplifying assumptions may leave out real-world factors.

AP COMPUTER SCIENCE PRINCIPLES • EXERCISE SHEET

3.17 Algorithmic Efficiency

Name: _____ Class: _____ Date: _____ **Total: 8 marks**

KEY WORDS heuristic 启发式 • efficiency 效率 • algorithmic efficiency 算法效率 • algorithm 算法

Objective

Build the skills to answer exam questions on **algorithmic efficiency**.

You must be able to:

- explain how **algorithmic efficiency** 算法效率 compares algorithms by the resources they use
- estimate the number of **steps** an algorithm takes as the input grows
- distinguish a **reasonable** from an **unreasonable** running time
- use a **heuristic** 启发式 when an exact solution is too slow

1 Worked examples

Study these first. Each one shows the method for a question type used later.

■ Efficiency

Compares algorithms by the **resources** (steps or time) they use as the input grows.

■ Reasonable vs unreasonable

A **reasonable** running time grows slowly (e.g. proportional to n or n^2). An **unreasonable** one grows explosively (e.g. 2^n), becoming impractical on large inputs.

■ Heuristics

When an exact solution is too slow, a **heuristic** finds a **good-enough** answer quickly.

2 Practice

2.1 State what algorithmic efficiency compares. [1]

2.2 State what a heuristic is. [1]

2.3 State why some correct algorithms are not used in practice. [1]

3 Exam-style questions

3.1 Algorithmic efficiency compares algorithms by [1]

- **A** their length in lines
- **B** the resources they use
- **C** their author
- **D** the language used

3.2 A heuristic gives a [1]

- **A** guaranteed exact answer
- **B** good-enough answer quickly
- **C** deliberately wrong answer
- **D** random answer

3.3 Algorithm A takes about n steps; algorithm B takes about 2^n steps.

(a) State which is more efficient for large n. [1]

(b) State whether B could be too slow to use. [1]

(c) State one option when an exact solution is too slow. [1]

Solutions

2.1 the resources (steps or time) that algorithms use.

2.2 an approach that finds a good-enough solution when an exact one is too slow.

2.3 they take an unreasonable (too long) time on large inputs.

3.1 B.

3.2 B.

3.3 (a) A. (b) yes. (c) use a heuristic.

AP COMPUTER SCIENCE PRINCIPLES • EXERCISE SHEET

3.18 Undecidable Problems

Name: _____ Class: _____ Date: _____ **Total: 8 marks**

KEY WORDS decidable 可判定 • undecidable 不可判定 • inefficient 低效 • algorithm 算法 • efficiency 效率

Objective

Build the skills to answer exam questions on **undecidable problems**.

You must be able to:

- explain how a **decidable** 可判定 problem has an algorithm that always gives a correct yes-or-no answer
- explain how an **undecidable** 不可判定 problem has no algorithm that solves every case
- describe the difference between undecidable and merely **inefficient** 低效
- recognize undecidability as a **limit** on computation

1 Worked examples

Study these first. Each one shows the method for a question type used later.

■ Decidable vs undecidable

- A **decidable** problem has an algorithm that **always** gives a correct yes-or-no answer.
- An **undecidable** problem has **no** algorithm that solves **every** case correctly.

■ Undecidable is not just slow

Undecidability is stronger than inefficiency: an inefficient problem can still be solved (given enough time), but an undecidable one **cannot** be solved for all cases at all.

■ A limit on computation

Undecidability shows there are things computers simply **cannot** do.

2 Practice

2.1 Define a decidable problem. [1]

2.2 Define an undecidable problem. [1]

2.3 State the difference between an undecidable problem and a merely inefficient one. [1]

3 Exam-style questions

3.1 A decidable problem has an algorithm that [1]

- **A** always gives a correct yes-or-no answer
- **B** never finishes
- **C** gives a random answer
- **D** is simply slow

3.2 An undecidable problem [1]

- **A** can always be solved slowly
- **B** has no algorithm that solves every case correctly
- **C** is a syntax error
- **D** is a data type

3.3 A problem has no algorithm that can correctly answer it in **all** cases.

(a) Name this kind of problem. [1]

(b) State whether this is the same as being slow. [1]

(c) State what it shows about computation. [1]

Solutions

2.1 a problem with an algorithm that always gives a correct yes-or-no answer.

2.2 a problem for which no algorithm can solve every case correctly.

2.3 an inefficient problem can still be solved (just slowly); an undecidable one cannot be solved for all cases at all.

3.1 A.

3.2 B.

3.3 (a) an undecidable problem. (b) no. (c) that there are limits to what computers can do.