HOLIDAY HOMEWORK 假期作业

AP Computer Science Principles

Exercise sheets and past-paper practice, topic by topic.

每个单元：练习卷 + 历年真题。

For class or self-study • no answer keys included.

Contents 目录

1 Designing programs – Part 1

Name: _____ Class: _____ Date: _____

Vocabulary 词汇

Write each word three times. 把每个单词抄写三遍。

English	中文	Write it 3 times (English)		
input	输入			
output	输出			
program	程序			

Recall

You have used many programs – apps, games, websites:

- Every program was designed by someone to do a job.
- Big programs are built by teams working together.
- A program takes something in and gives something back.

This sheet lines up how programs are made. Each idea has a worked example.

New ideas

1 Input, process, output

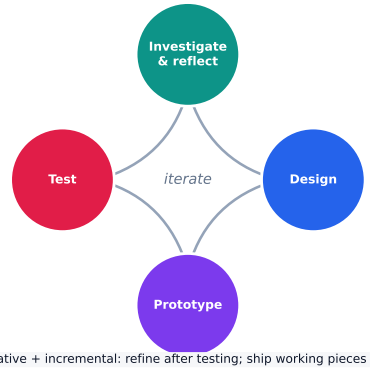
Most programs follow the **input**–**process**–**output** pattern: take an **input** 输入, do some processing, then give an **output** 输出.

computing system model: Input → Process → Output (IPO)

Worked example. A calculator app takes two numbers (input), adds them (process), and shows the total (output).

2 The development cycle

A **program** 程序 is built step by step: plan it, write it, test it, and improve it – often looping back to fix problems.



iterative + incremental: refine after testing; ship working pieces early

3 Working together

Large programs are built by **teams**. Collaboration – sharing ideas and reviewing each other's work – produces better, more reliable programs.

4 Purpose

Every good program has a clear **purpose** – a problem it solves or a need it meets. Knowing the purpose guides every design decision.

Watch out: writing code is only one part of making a program. Planning what it should do, and testing that it really works, matter just as much – skipping them leads to buggy software.

Practice

Try these in order. The methods are all above.

2.1 State the three parts of the input–process–output pattern. [3]

2.2 For a program that finds the average of some test scores, state its input, process, and output. [3]

2.3 State two things that happen in the program development cycle besides writing code. [2]

2.4 State one benefit of programmers working in a team. [1]

2.5 Explain why knowing a program's purpose is important. [1]

1. Identify an example output of your program. Explain how this output shows an aspect of your program's functionality.

1 Errors and debugging – Part 2

Name: _____ Class: _____ Date: _____

Vocabulary 词汇

Write each word three times. 把每个单词抄写三遍。

English	中文	Write it 3 times (English)		
bug	错误			
Debugging	调试			

Recall

In Part A you saw that programs are tested. Testing finds **bugs**:

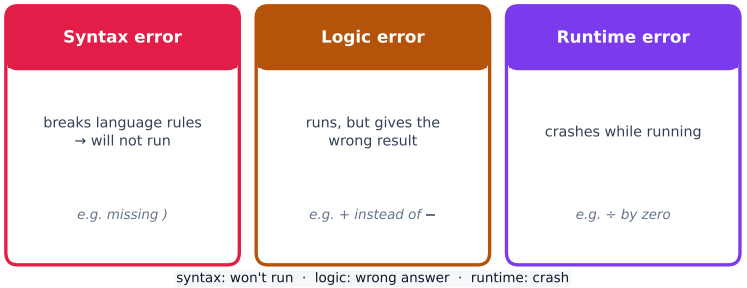
- A "bug" is a mistake in a program.
- Even careful programmers make them.
- Finding and fixing bugs is a normal part of coding.

Each idea has a worked example before the Practice.

New ideas

1 What a bug is

A **bug** 错误 is an error in a program that makes it behave wrongly. There are a few kinds:



- a **syntax error** (breaking the language's grammar, so it won't run),
- a **logic error** (it runs but gives the wrong answer),
- a **runtime error** (it crashes while running).

2 Debugging

Debugging 调试 is the process of finding and fixing bugs – by testing, reading the code, and checking values step by step.

3 Trace tables

A **trace table** follows a program's variables line by line, so you can see exactly where it goes wrong.

count	total	output
1	5	
2	12	
3	20	20

4 Testing

Good testing tries many inputs – ordinary ones, edge cases (like 0 or an empty list), and wrong inputs – to catch bugs before users do.

Watch out: a program with **no error message** can still be **wrong** – a logic error runs fine but produces the wrong answer. That is why you must check the output, not just that it runs.

Practice

Try these in order. The methods are all above.

2.1 State what a bug is. [1]

2.2 State the difference between a syntax error and a logic error. [2]

2.3 State what "debugging" means. [1]

2.4 State what a trace table is used for. [2]

2.5 Explain why a program that runs without any error might still be wrong. [2]

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

1. Identify an unexpected or invalid input that a user could provide to your program. Describe the behavior of your program after it receives this input. If it is not possible for your program to accept an unexpected or invalid input, explain why this is the case.

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 Binary numbers – Part 1

Name: _____ Class: _____ Date: _____

Vocabulary 词汇

Write each word three times. 把每个单词抄写三遍。

English	中文	Write it 3 times (English)
Binary	二进制	

Recall

Computers store everything using only two symbols:

- Deep down, a computer uses only `0`s and `1`s.
- Text, pictures, and sound are all stored as numbers.
- You may have heard of "bits" and "bytes".

This sheet lines up binary numbers. Each idea has a worked example.

New ideas

1 Bits and bytes

A **bit** is a single `0` or `1`. A group of 8 bits is a **byte**. Everything a computer stores is made of bits.

2 Binary place values

Binary 二进制 (base 2) uses place values that are powers of two: $\ldots, 16, 8, 4, 2, 1$ – instead of the $\ldots, 100, 10, 1$ of our usual base 10.

place value	128	64	32	16	8	4	2	1
bits	**1**	0	0	**1**	0	**1**	**1**	0

$150 = 128 + 16 + 4 + 2 \Rightarrow$ `10010110`

3 Binary to decimal

To convert, add the place values that have a `1`.

Worked example. Binary `1101` $= 8 + 4 + 0 + 1 = 13$.

4 Decimal to binary

Subtract the largest power of two that fits, and repeat.

Worked example. 19: take 16 (leaves 3), then 2 (leaves 1), then 1. So the `16`, `2`, and `1` places are on $\rightarrow$ `10011`.

Watch out: a computer has a **limited** number of bits, so it can only store numbers up to a point. A number too big to fit causes an **overflow** error.

Practice

Try these in order. The methods are all above.

2.1 State how many bits are in a byte. [1]

2.2 List the first five binary place values, from the right. [1]

2.3 Convert binary `1010` to decimal. [2]

2.4 Convert binary `1111` to decimal. [2]

2.5 Convert the decimal number 6 to binary. [2]

2 Compression and information – Part 2

Name: _____ Class: _____ Date: _____

Recall

In Part A you saw that data is stored as bits. Now how we make data smaller and useful:

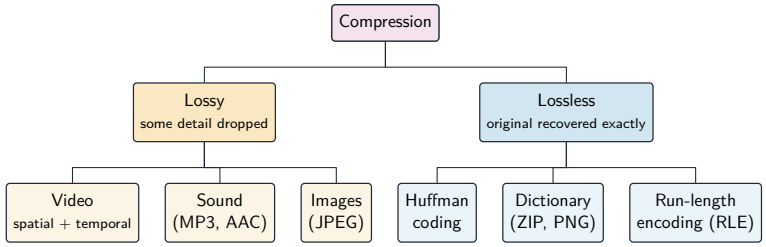
- You may have "zipped" files to make them smaller.
- Photos and music come in different quality settings.
- Data becomes useful when we find patterns in it.

Each idea has a worked example before the Practice.

New ideas

1 Compression

Compression reduces the number of bits needed to store data. There are two kinds:



- **lossless** – you can restore the **exact** original (used for text and programs),
- **lossy** – some data is thrown away to shrink it more (used for photos, music, video).

2 The trade-off

Lossless keeps everything but saves less; lossy saves more but loses some detail permanently. Choose lossless when the data must be exact.

3 Data to information

Raw **data** becomes useful **information** when we find patterns, trends, and answers in it – often using programs to sort, filter, and chart it.

4 Correlation is not causation

If two things rise together, that does **not** prove one caused the other. A pattern in data is a clue, not a proof.

Watch out: lossy compression **permanently** removes data – you cannot get the exact original back. Use it only when a small loss of quality is acceptable, never for data that must stay exact (like a program's code).

Practice

Try these in order. The methods are all above.

2.1 State the difference between lossless and lossy compression. [2]

2.2 State which type of compression you would use for a text document, and why. [2]

2.3 State which type is normally used for photos and music. [1]

2.4 State what turns raw data into useful information. [2]

2.5 Two things rise together in some data. Explain why this does not prove one caused the other. [2]

3 Variables, algorithms, and conditionals – Part 1

Name: _____ Class: _____ Date: _____

Vocabulary 词汇

Write each word three times. 把每个单词抄写三遍。

English	中文	Write it 3 times (English)
algorithm	算法	

Recall

Programs follow instructions to solve a problem:

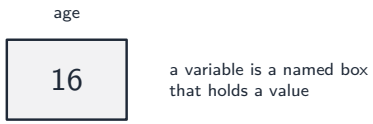
- They store values and calculate with them.
- They make choices depending on the situation.
- A step-by-step method for solving a problem is an algorithm.

This sheet lines up the basics of programming. Each idea has a worked example.

New ideas

1 Variables

A **variable** is a named store that holds a value, which can change as the program runs.



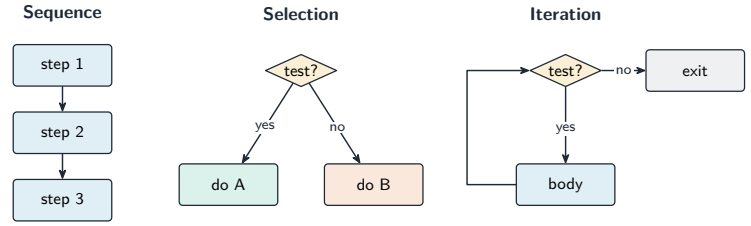

```
score <- 0
score <- score + 10   // score is now 10
```


2 Expressions

An **expression** combines values and operators to compute a result, such as `price * quantity`.

3 Algorithms

An **algorithm** 算法 is a step-by-step method to solve a problem – like a recipe. It uses three building blocks: sequence, selection (choices), and iteration (repeats).



4 Conditionals

A **conditional** (IF) chooses what to do based on a true/false test:

```
IF (score >= 50)
    DISPLAY "Pass"
ELSE
    DISPLAY "Fail"
```

Watch out: the order of steps matters. `score <- 0` then `score <- score + 10` gives `10`; swapping the lines would add `10` to an unset value. Sequence is the first building block of every algorithm.

Practice

Try these in order. The methods are all above.

2.1 State what a variable is. [2]

2.2 After `x <- 5` then `x <- x + 3`, state the value of `x`. [2]

2.3 State what an algorithm is. [2]

2.4 Given `score = 40` and the IF/ELSE above, state what is displayed. [1]

2.5 Name the three building blocks of an algorithm. [3]

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 `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

Name: AP Computer Science Principles: **2025 Set 2**

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 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

3 Iteration, lists, and efficiency – Part 2

Name: _____ Class: _____ Date: _____

Recall

In Part A you met variables and choices. Now repeating steps and handling many values:

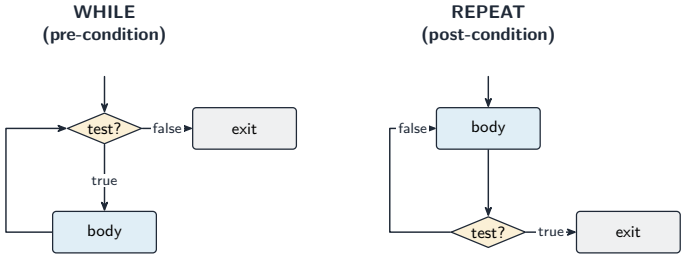
- Doing something 100 times by hand is slow; a loop does it fast.
- A list holds many values under one name.
- Some methods are much faster than others.

Each idea has a worked example before the Practice.

New ideas

1 Iteration

Iteration (a loop) repeats steps. `REPEAT 5 TIMES` runs its block five times; a conditional loop repeats while a test stays true.



tested first ⇒ may run **zero** times

tested last ⇒ runs **at least once**

2 Lists

A **list** stores many values in order under one name, each reached by its position (index).

3 Binary search

To find a value in a **sorted** list, **binary search** checks the middle and throws away the half that cannot contain the target, repeating.

Worked example. A sorted list of 8 items needs at most 3 checks ($8 \to 4 \to 2 \to 1$), far fewer than checking all 8.

4 Efficiency

An algorithm's **efficiency** is how its work grows with the input size. A **reasonable-time** method grows slowly enough to be practical; an **unreasonable-time** one grows so fast it becomes impossible for large inputs.

Watch out: binary search is only fast because it **halves** the list each step. It only works on a **sorted** list – on an unsorted one you must use the slower linear search.

Practice

Try these in order. The methods are all above.

2.1 State what iteration (a loop) does. [1]

2.2 State what a list is. [2]

2.3 A sorted list has 16 items. State roughly how many checks binary search needs. [2]

2.4 State the one requirement binary search has of the list. [1]

2.5 Explain the difference between a reasonable-time and an unreasonable-time algorithm. [2]

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 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 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 `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

4 The internet – Part 1

Name: _____ Class: _____ Date: _____

Recall

You use the internet every day:

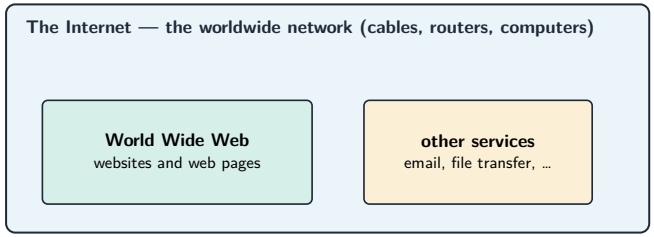
- Websites, messages, and videos all travel over the internet.
- The internet connects computers all over the world.
- Somehow your message reaches exactly the right place.

This sheet lines up how the internet works. Each idea has a worked example.

New ideas

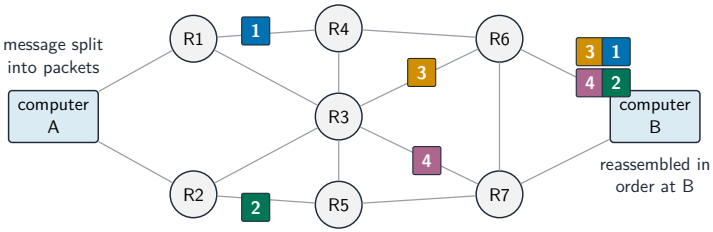
1 The internet

The **internet** is a global **network of networks**. The **web** (websites) is just **one service** that runs on it – along with email, video calls, and more.



2 Packets

Data is broken into small **packets** that are sent separately and reassembled at the destination.



3 Protocols

A **protocol** is an agreed set of rules for communication – like a shared language that lets different computers understand each other.

4 Fault tolerance

Because there are **many** possible paths between two points, if one path fails, packets take another. This makes the internet **fault-tolerant** – it keeps working even when parts break.

Watch out: the **internet** and the **web** are not the same thing. The internet is the worldwide network of connections; the web is just one of the services that runs on top of it.

Practice

Try these in order. The methods are all above.

2.1 State the difference between the internet and the web. [2]

2.2 State what a packet is. [2]

2.3 State what a protocol is. [2]

2.4 Explain why the internet keeps working even when one connection fails. [2]

2.5 State what "fault-tolerant" means. [1]

4 Parallel and distributed computing – Part 2

Name: _____ Class: _____ Date: _____

Recall

In Part A you met the internet. Now how computers work **faster** by sharing the job:

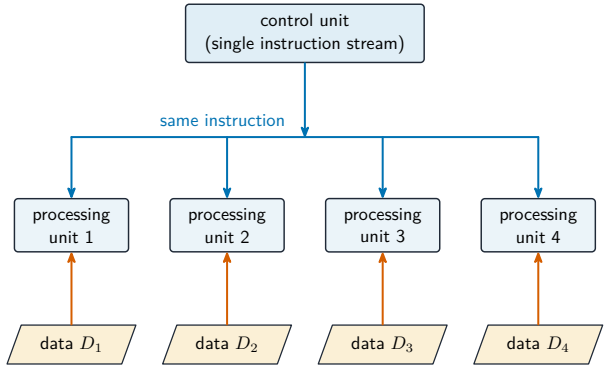
- Many hands make light work – computers can split a job too.
- Big problems are solved by many computers together.

Each idea has a worked example before the Practice.

New ideas

1 Sequential versus parallel

- **Sequential computing** does one step at a time.
- **Parallel computing** splits a task into parts that run **at the same time** on several processors, finishing sooner.



2 Speedup

The **speedup** is how much faster the parallel version is:

$$\text{speedup} = \frac{\text{sequential time}}{\text{parallel time}}.$$

Worked example. If a job takes 100 seconds one at a time but 40 seconds split up, the speedup is $\frac{100}{40} = 2.5$.

3 The limit

Speedup is limited: parts that **must** run in order cannot be sped up by adding more processors, so doubling the processors rarely doubles the speed.

4 Distributed computing

Distributed computing uses **many computers** over a network to solve one problem too big for a single machine.

Watch out: the part of a task that **must** run in sequence sets a floor on how fast the whole job can finish. Adding more processors only speeds up the parallel part, not the sequential part.

Practice

Try these in order. The methods are all above.

2.1 State the difference between sequential and parallel computing. [2]

2.2 A task takes 60 seconds sequentially and 20 seconds in parallel. Find the speedup. [2]

2.3 A task takes 80 seconds sequentially and 32 seconds in parallel. Find the speedup. [2]

2.4 State why doubling the processors rarely doubles the speed. [2]

2.5 State what distributed computing is. [1]

5 The impact of computing – Part 1

Name: _____ Class: _____ Date: _____

Recall

Computing has changed almost everything:

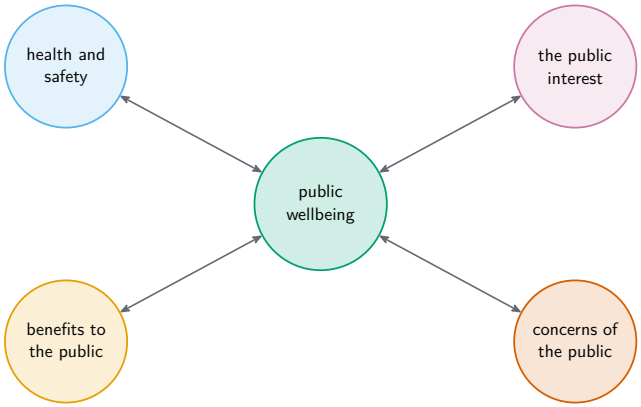
- Phones and the internet connect people across the world.
- But not everyone has the same access to technology.
- Computing brings both benefits and problems.

This sheet lines up the effects of computing on society. Each idea has a worked example.

New ideas

1 Benefits and harms

Computing brings big **benefits** (fast communication, medical advances, access to knowledge) but also **harms** (loss of privacy, misinformation, job changes).



Most computing innovations have **both** – the same tool can help and harm.

2 The digital divide

The **digital divide** is the gap between those who have good access to computers and the internet and those who do not. It can come from income, location, or age, and it can widen other inequalities.

3 Crowdsourcing

Crowdsourcing gathers help or ideas from a large group of people online – for example, mapping disaster areas or funding a project. It puts computing's connectivity to good use.

4 Thinking about impact

When judging a new technology, ask **who benefits**, **who is harmed**, and **who is left out** – not just whether it works.

Watch out: almost every computing innovation has **both** good and bad effects. The same social media that connects friends can also spread misinformation – so impact is rarely all positive or all negative.

Practice

Try these in order. The methods are all above.

2.1 Give one benefit and one harm of computing. [2]

2.2 State what the "digital divide" is. [2]

2.3 State one cause of the digital divide. [1]

2.4 State what crowdsourcing is, with an example. [2]

2.5 Explain why most technologies have both good and bad effects. [2]

5 Bias, privacy, and safe computing – Part 2

Name: _____ Class: _____ Date: _____

Recall

In Part A you met the effects of computing. Now fairness and keeping data safe:

- Programs can be unfair if they learn from unfair data.
- Your personal information needs protecting.
- Messages can be scrambled so only the right person can read them.

Each idea has a worked example before the Practice.

New ideas

1 Computing bias

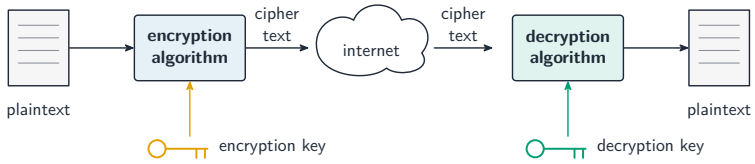
A program can carry **bias** if the data it learns from does not represent everyone fairly – leading to unfair results for some groups. The bias comes from the data and design, not the machine "deciding" to be unfair.

2 Privacy

Programs that collect data raise **privacy** concerns. Good practice: collect only what is needed, protect it, and be honest about how it is used.

3 Encryption

Encryption scrambles data so that only someone with the right key can read it – keeping messages and passwords safe as they travel over the internet.



■ 4 Safe computing

Protect yourself with strong, unique passwords, caution with unknown links and downloads, and keeping software up to date.

Watch out: computing bias usually comes from **biased data**, not a machine choosing to be unfair. If a program is trained on data that under-represents some people, it can treat them unfairly – so the fix is better data and design.

Practice

Try these in order. The methods are all above.

2.1 State where computing bias usually comes from. [2]

2.2 State two good practices for handling people's personal data. [2]

2.3 State what encryption does. [2]

2.4 Give two ways to stay safe online. [2]

2.5 Explain why a program trained on unfair data can give unfair results. [2]