Zig Zag Education

Computer Science

ZigZag's 50 Algorithm Challenges with Python Solutions

for programming students

- DIGITAL RESOURCE -

This pack includes paper versions of the electronic files.

Go to zzed.uk/productsupport to download the electronic files.

zigzageducation.co.uk

POD 12860

Contents

Teacher Introduction

Through these tasks, beginner and intermediate programming students develop their skills to the level where they create a mini-games website. The exercises are arranged in increasing difficulty for students to complete in order. Intermediate students should find the initial exercises a very quick refresher. These challenges are also suitable for practising drawing flow diagrams and creating pseudocode solutions.

Although not essential, I recommend that students work through these in order, as most of the challenges use programming techniques learnt in previous challenges. If students are stuck on some of the more involved challenges, then one option is to provide them with the pseudocode solutions. You may want to leave out a couple of the more advanced challenges, such as the maze creator trees and graphs marked '(harder)', for all but the most motivated students below Year 12.

Also see the advice to students about using AI, in the student notes.

Please do send us suggested improvements – if we make changes as a result we will send you an updated copy.

Solutions, Pseudocode and Flowcharts

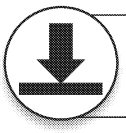
The source code for the solutions to all 50 algorithms is provided in Python 3.10. See details below to download them. It also includes a folder for playing cards images to be used for the 'Pairs' and 'Higher or Lower' game challenges.

Pseudocode solutions are provided to all 28 basic and common practice tasks in sections 1 and 2, and to the Minesweeper game algorithm. You could set your students to write the pseudocode for the tasks before they start coding.

Flowcharts are provided for the four sorting algorithms; again, students could be asked to create the flowcharts before they start coding, or to work through them to understand how they work, or they could be asked to convert the flowcharts to pseudocode.

There will be multiple ways to solve longer challenges. Credit should be given for working solutions that meet the requirements. Students that finish tasks earlier than others can be challenged to compare solutions with others and to find the most efficient solution, as this is also an important skill to develop.

November 2025



Electronic files for this resource are provided on the ZigZag Education Support Files system, which can be accessed via **zzed.uk/productsupport**

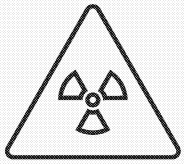
Student Introduction

This set of tasks is intended for you to learn through doing. It can be used by someone with no experience of programming. If you already have some programming experience you should still do all the tasks but will find them super-quick to do, until you find your level.

It is important to add comments and indentation to make your code easier to read. At the end of each task compare your code to the example solution. There is rarely a 'best' solution, but when you compare your code to other students' code and to the example solution, think about a balance of the following points:

- Which code is right and always gives the right result?
- Which code caters for erroneous input and never crashes?
- Which code runs the fastest?
- Which code is the easiest to read, well-structured and with concise comments where appropriate, and therefore is easiest for someone else to make changes?
- Which code didn't take too long to write?

Of course, many of the algorithms can be found on the Internet, and AI sites such as ChatGPT and Gemini will write solutions to most of them. However, resist using them as it defeats the object of you learning to code and gaining experience of coding. Once you are experienced at coding you can then learn how to use AI to greatly speed up your coding, along with the different ways to use it, and its weaknesses.



If your teacher allows you access to AI, we recommend you use it in these ways:

1. When you are struggling to debug a line – ask the AI what is wrong with it.
2. When you have finished and tested your program, ask the AI if there are any problems with your code and how it could be improved.
3. Ask the AI how your code can be optimised to make it faster.
4. Ask the AI how your code can be made more readable.

Create a menu to link to each of your solutions. If you are using a web-based programming language such as PHP, then create an HTML index page with links to your solutions and at the end of each solution add a link back to the menu.

Section 1 – The Basics

This collection of 18 short tasks covers all the basic programming constructs. If y
be able to do all of these, using any basic programming book or website for the l
the *cheat sheet* at the end of this resource). If you already have experience of c
whizz through this section at high speed.

Either way, take a small amount of extr[illegible]o:

1. Ensure your code is readable w[illegible]ndentations, spacing and the occasiona
2. Give clear user ir[illegible]on the screen and present things clearly.
3. Test [illegible]od[illegible]cater for erroneous user input.

Task 1.1 – The RANDOM Func

Task

Output each of the following on a new line:

- A random whole number from 0 to 10 inclusive
- A random whole number from 50 to 100 inclusive
- A random number from 0 to 10 inclusive, to 2 decimal places

You should find that each time you run the program you get different number

Hints

- If you get the same number every time then you [illegible] programming language wh
  random numbers.
- Some languages automati[illegible]ou[illegible]new line (sometimes called a return), e.g. in Py
  automatically pri[illegible]w[illegible]e. Some languages like VB.NET have an option for with and
  pri[illegible]li[illegible]but *Write()* does not. In some languages, like JavaScript, you put '\n' a
  wa[illegible]line. Finally, if your output is in HTML (e.g. from PHP) then you can either u
  the paragraph in <P>...</P>.

Task 1.2 – Variables

Task

A variable holds data, and it can change:

- Put a random number from 0 to 10, inclusive, into a variable called *rando*
- Put a random number from 1,000 to 2,000, inclu[illegible]e, into a variable calle
- Output both variables and then output th[illegible]to[illegible]of the two variables adde
  new line.

Task 1.3 – Concatenation

Task

Store the strings "super", "fast", "fly" in three variables, concatenate them with spaces between them, and output the result.

Notes

Concatenation means joining strings together; for example, concatenating the strings "house" and "boat" creates the string "house boat". Alternatively, if you concatenate the strings "house", " ", and "boat", this creates the string "house boat".

Hints

- PHP concaten
  e.g. *echo "hou*
- Python concat
  e.g. *print("hou*
- JavaScript als
  e.g. *console.lo*

In some languages you can concatenate different types of variables together, e
Booleans. In other languages you have to convert other data types to strings b
concatenated.

Task 1.4 – Arithmetic

Task

Generate a random number between 1 and 100 inclusive, and another betwee
with an explanation of what each number is:

- The two numbers added together
- The second number subtracted from the first number
- The two numbers multiplied with th
- The first number to the power of the second number
- The first number divided by the second number
- The number and remainder when the first number is divided by the

Task 1.5 – FOR Loop

Task

Output the numbers 1 to 10 using a *FOR loop*, on one continuous line, with a space between each number.

Task 1.6

Task

Output the 17 time
1,000, using a *WHI*

Task 1.7 – DO UNTIL Loop

Task

The Fibonacci series beg , 2, 3, 5, 8, …

It starts an , then each subsequent number is the sum of the previous two numbers.

Output the Fibonacci series using a *DO UNTIL loop*, until you get to over 1,000.

Task 1.8

Task

Iterate through (lo
from 1 to 1,000. I
both 5 and 7 then
an asterisk. Outpu
continuous line.

Task 1.9 – Boolean Operato

Task

A student has taken two tests out of 50. Generate two random test scores bet

- If either one of the scores is above 30 then output "Well done, you passe
- If both scores are above 30 then output "You are amazing, you have passe
- If neither score is above 20 then [illegible] output "Simply NOT good e

Run your code repeat[illegible] you have seen all outcomes.

Task 1.10 – CASE/SWITC

Task

Generate a random number from 1 to 7. Use a case statement to output whether it is a weekday, a Saturday or a Sunday, where 1 = Monday, 2 = Tuesday, etc.

Hint

Some languages cal
switch statement. I
match-case stateme

Task 1.11 – String Manipulat

Task

Using the two real words *uncharacteristically* and *hippopotomonstrosesquiped*

1. Combine the first 5 characters of the second word and the last 5 characte
2. Combine the first 5 characters and the last 5 characters of the second wo
3. Co
   a. [illegible]aracters from the first word starting from the 3rd character with
   b. 5 characters from the second word starting from the 15th character.
4. Combine:
   a. 4 characters from the second word starting from the 20th character w
   b. 6 characters from the first word starting from the 12th character and
   c. 4 characters from the end of the second word.

Print out the two original words and the four new words.
Calculate and display in brackets, after each of the six words, how many characters are in each word.

Hint

Most pro
start wit

Task 1.12 – Arrays

Task

Here is an array of numbers: [34, 7, 12, 45, 99, 2, 88, 66, 23, 18, 56, 9, 91, 42]

Your task is to:

1. Separate the odd and even numbers into two different arrays.
2. Display how many numbers are in each of the arrays.
3. Display the contents of the odd and even numbers.
4. Sort the odd and even arrays into ascending order and display them.

Hints

- To work out if a number is odd or even, calculate the remainder when dividing by 2 and see if it equals 0. Store the remainder in some language variable.
- In most programming languages, an array index goes from *0* to *n-1*.

Task 1.13 – 2D Arrays

Task

Here is a classroom seating arrangement.

Row 1	Aye
Row 2	Jam
Row 3	The
Row 4	Sop

Create a 2D array that represents this classroom seating arrangement, where each row in the array corresponds to a row of seats in the classroom, and each column corresponds to a seat in that row.

- Display the seating arrangement with student names.
- Count and display the number of students, the number of empty seats, and the total number of seats in the classroom.
- Display a linear version of the seating arrangement of the names with row and column numbers.

Notes

In Python, you can create a 2D array like this:

```
matrix = [
   [1, 2, 3],
   [4, 5, 6],
   [7, 8, 9]
]
```

Rows and columns start at 0. You can output any item by referencing its row and column, e.g. *print(matrix[1][1])* would output 5.

Task 1.14 – Character Cod

Task

Create a random password made up of:

1. 4 random capital letters
2. 4 random numbers
3. 4 random lower-case letters
4. 4 random characters from an array of these characters: £, $, %, ^, & and *

Display the random password, then convert and display the same random pass
lower case.

Hints

- a–z are ch
- A–Z are ch

Notes

Example Python commands for manipulating strings:

```
sentence = "hello WORLD"
print(sentence.lower())  # Output: hello world
print(sentence.upper())  # Output: HELLO WORLD
print(sentence.capitalize())  # Output: Hello world
print(sentence.title())  # Output: Hello World
print(len(sentence))  # Output: 11
print(sentence.replace(" ", ""))  # Output: helloWORLD
```

Task 1.15 – Procedures

Task

Create a procedure which outputs the information from any 2D array in an HT
shaded heading row (no matter the size of the array).

Time Slot	Monday	Tuesday	Wednesday	Thursday	Fri
9:00 - 10:00	Maths	English	Science	History	Art
10:00 - 11:00	Physics	Biology	Maths	English	PE
11:00 - 12:00	Chemistry	Geography	Physics	Art	Sci
12:00 - 1:00	Lunch	Lunch	Lunch	Lunch	Lun
1:00 - 2:00	History	Science	PE	Maths	Eng
2:00 - 3:00	Art	History	Biology	Geography	Phy

Create an array of the information above and pass it into your procedure.

If your code is not being run in a web browser, then save the output to an HTML file and view it in a browser to check your output.

Hint

In some langua
function that d

Notes

Example Python proce

```
def print_stars(x):
    #Print x asterisks in a row.
    print("*" * x)

# Example usage:
print_stars(5)   # Output: *****
print_stars(10)  # Output: **********
```

- ***def*** *is used to define a proce*
- ***x*** *is a parameter — it takes i*
- *Note how * (the multiply sy*
  strings in Python ("" * 5 →*

Task 1.16 – Functions

Task

In maths, 5! means 5 factorial, which means 5 × 4 × 3 × 2 × 1 = 120.

- Create a function called *factorial1* which is given a number and returns th using a loop.
- Create a recursive function called *factorial2* to work out and return the fa
- Write a program which works out 1 using both methods and displays (to 6 decimal places).

Computers are getting increasingly powerful, and different languages will do this with more or less efficiency. If 170! takes is too big to display, or is too quick to register a result, then increase the number.

Notes

Example Python function:

```
def greet(name):
    return "Hello, " + name + "!"

# Call the function
greeting = greet("Alice")
print(greeting)
greeting = greet("Bob")
print(greeting)
```

- ***def*** *is used to define a procedure or*
- ***name*** *is a parameter — it takes inp*
- *The function is called using its nam*

Task 1.17 – Files

Task

Create a secret message to a friend, then:

- Write a function to encode any string into a coded message by reversing the string.
- Write a function to decode the string.
- Save the secret message to your friend in a variable.
- Encode the message and save the coded message to a file called 'top-secret.txt'.
- Read the file back in, decode it, and output it.
- Check that the file has been properly created.

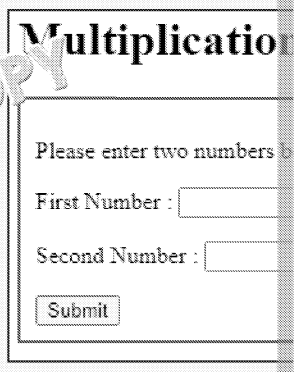
Hint

Ensure you ou message if th happen if the don't have th

Task 1.18 – Input

Task

- Prompt the user to type two numbers.
- Ensure they are integers and multiply them together.
- Output the result.



Notes

You can use the command line for input, or you can use a form to input numbers with two input text boxes and one button, like the example shown here.

Hint

You will need to check that the input fields are not empty.

Section 2 – Common Practic

Each of the following exercises are common challenges to face and are an oppo
programming constructs from section 1.

Task 2.1 – Hex Colo

Task

An HTML c m e up of three colours: red, green and blue. Each colour is
number fro F (from 0 to 15 in decimal), where 10 is A, 11 is B, 12 is C, 13 is
colours are, therefore, written as hexadecimal numbers from #000000 to #FFFFF

Output a table that displays all the colours where each of the three colours is a d
#110000, #220000, up to #FFFFFF.

Notes

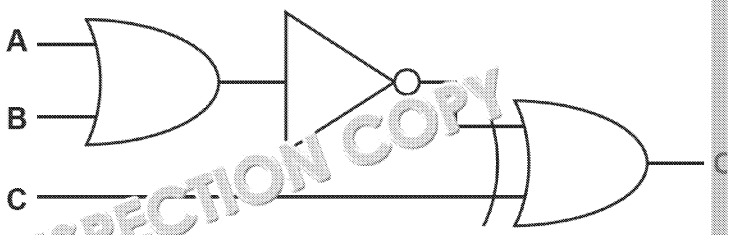
- Create a function that takes in three integers from 0 to 15 and converts the
  number with a # at the front.
- Table cells can have a background colour which is set with the hexadecimal
- Rotate through all the possible combinations of x, y and z being the numbers
  hexadecimal using your function, and output them in a table cell with the ba
  hexadecimal number.
- Use a CSS <style> command to create styling for your t e cells.

If your code is not being run in a web brow th ave the output to an HTML
check your output.

Hi

- Use three nested loops to loop through all the different combinations.
- You will need to know or look up how to create HTML tables for this task
- If coding in PHP or other web-based language, you don't need to use a f

Task 2.2 – Logic Gates



Task

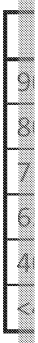
Output the bl or the logic circuit shown above.

Task 2.3 – What Grade Did I Get?

Task

- Create a function called *printgrade* which is passed the percentage and returns a sentence that says 'Grade x achieved', where x is the grade, based on the following table. It should accept the input as a decimal number, e.g. 78.5.

- Ask the user to enter their test exam mark out of 100 and then give them their resulting grade.



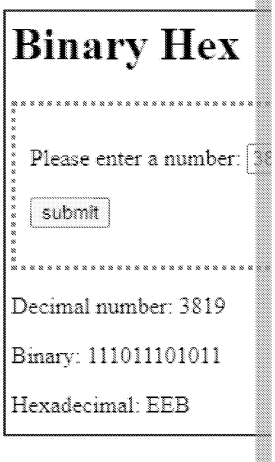
Hints

- Don't use multiple *if* statements to calculate the grade; use a *case* statement if the lang has it. For example, PHP has a *switch case* statement and Python has a *match* statemen
- When your code is working, do some testing by typing in marks in each grade range an check it works perfectly.
- Even if you limit your text box to accept numbers only, it is still a good idea to check tha that if something goes wrong it doesn't crash your program.

Task 2.4 – Binary Hex

Task

Ask the user to type in a decimal number. Work out and output the number as a binary number and as a hexadecimal number.



Manually convert the decimal to binary with your own algorithm; do not use an inbuilt function, otherwise you are not learning all the skills you need. (You can, however, use inbuilt function, if there is one, to convert from binary to hexadecimal.)

Test your program with 3819 and check you get the same results as shown here.

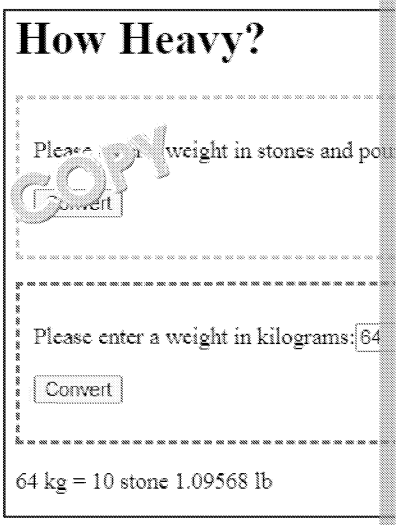
Hints

- Most programming languages have a function to convert from binary to hexadecimal use. For example, in PHP you can use the inbuilt *hexadecimal()* function. In Python *hex(int(binary, 2))[2:].upper()*, which converts the binary string to an integer, and the
- Plan what extreme and erroneous data might be entered and cater for these in your p not both 2,000,000,000 and 3,000,000,000 give the right answers.

Task 2.5 – Weight Converter

Task

Ask the user to choose between entering a weight in stones and pounds to display in kilograms or entering a weight in kilograms to display in stones and pounds.



Hints

- 10 kg = 2.0 [illegible] =
  1 s[illegible] 62 pounds
- Note that there are 14 pounds in a stone
- You will need to use the MOD function
- In the attached web-based screenshot, the program will need to know which 'Convert' button has been pressed

Task 2.6 – How Much Change?

Task

Assuming a customer has paid with a £20 note, and that they are purchasing an it the user to enter the cost of the item and calculate the change the customer shou individual notes and coins.

For example, if an item cost [illegible] then the change from a £20 note would be £1 £5 note, a 5p coin a[illegible] coin.

Hints

- If the total change is more than £10 then they will get a £10 note, which yo subtract from the change and then repeat this check.
- There might be more than one coin of the same value; for example, an item would have 2 × 2p in the change.
- This task is a little trickier than it looks on the surface to get the output to co pounds or pence, and to indicate whether it is a note or a coin.

Task 2.7 – Table Transposition

Task

Set a constant SIZE to be 10. Create a 10 × 10 array of random numbers between
an HTML table. Then, flip the array by transposing the rows to columns and the c
again. For example:

9	3	1	6	3	5	4	6	2	
5	1	0	7	3	6		1	0	3
6	2				7	4	9	0	0
7	0	3	3	9	2	2	6	8	8
8	7	6	5	5	6	8	7	2	1
1	5	5	4	6	1	0	8	9	2
0	6	6	4	8	5	4	1	1	1
7	9	0	0	7	6	3	4	9	1
4	9	7	9	2	0	9	6	8	7
4	5	2	5	6	8	5	2	7	2

→

9	5	6	
3	1	2	
1	0	0	
6	7	1	
3	3	0	
5	6	7	
4	5	4	
6	1	9	
2	0	0	
3	3	0	

Hints

- Create a function to generate th a ra
- Create a procedure to ar n × n array.
- Create an s n to flip an *n* × *n* array; each coordinate (x,y) in the first grid be
- re using a programming language with web output, then create a procedure
  able.

Task 2.8 – Quadratic Solver

Task

In GCSE maths you have to solve quadratic equations in the form $ax^2 + bx + c =$

For example, solve $2x^2 - 3x + 17 = 0$

One method of solving th[illegible] the formula $x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$

In this exam[illegible] *2*, *b = -3* and *c = 17*

Write a program that asks a student for the values of a, b and c, and works out tw[illegible]

Hints

- Remember that square root normally gives two answers, which is why the formula ha[illegible] the square root of 4 is both -2 and +2.
- If $b^2 - 4ac = 0$, for example if $a = 1$, $b = 2$ and $c = 1$, then there is only one solution.
- You cannot square root a negative number. For example, if $a = 1$, $b = 1$ and $c = 4$, th[illegible] Therefore, the program needs to check for this and tell the user if it occurs.

Task 2.9 – Email Check[illegible]

Task

Ask the use[illegible]ou[illegible] email address and output whether or not it is a valid e[illegible] email addre[illegible]lain why.

Here are the rules of what is and isn't allowed in an email address:

- An email address is made up of the local part, followed by @, followed by th[illegible]
- The local part must be at least one character including upper- and lower-case[illegible] special characters *period (dot)*, *hyphen*, *underscore*, although the special cha[illegible] beginning or the end or appear consecutively (h.-h is allowed, but not h..h).
- The domain part must be a combination of alphanumeric characters and hyp[illegible] The hyphens and periods cannot appear at the beginning or end. There mus[illegible]
- The domain part must be between 2 and 255 characters, and the overall em[illegible] of 320 characters.

Task 2.10 – Maze Creator (harder)

Task

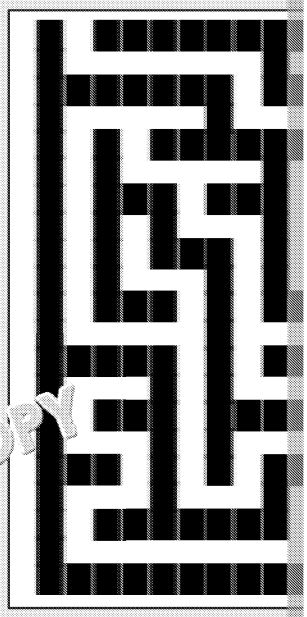
Generate a random maze, that has one solution, based on a grid specified by the user, that is from 11 × 11 to 49 × 49; the height and width can be different fro
the *recursive backtracking* logic below on paper first bef u start coding.

Hints

- Only use odd dimensions so the paths and walls align correctly.
- Start by making s re a block (wall).
- Start a , 1) so that (0,1) is part of the wall, and end the wall at the bottom right
  width
- Always move **2** steps when creating the maze (so a path doesn't touch another path).

Use *recursive backtracking* to create the maze:

1. Pick a random direction (Right, Left, Down, Up).
2. Check if the next cell (2 steps away) is inside the grid and is a wall (a block).
3. If it is a block, then carve a path by:
   - Converting the target cell into a path (' ').
   - Converting the wall between the target cell and the previous cell into a path (' ').
4. Recursively continue from the new position.
5. Backtrack (end the recursive function) when no more moves are possible.



Section 3 – Standard Computing

The following algorithms will be found on most GCSE, A Level, BTEC and OCR Tec
quite likely you will have an exam question based on one of these, so it is a good
listed on the specification you are learning without support, and to understand f
struggling to understand how any of them work then as[illegible]end who does to ex
you will find many YouTube explanations too.

Tasks 5 and 8 are linked to the c[illegible]f the efficiency of different algorithms,
computing specificatio[illegible]

Task 3.1 – Bubble Sort

Generate an array that contains 50 random integers between 1 and 100. It is fine
are repeated.

Write a program to put the array in size order using the Bubble sort algorithm, wh
comparing each number with the next, and swapping them over if the first numbe
until all the numbers are in order.

Task 3.2 – Merge Sort

Generate an array that contains 50 random integ[illegible]et [illegible]en 1 and 100. It is fine
are repeated.

Write a program to put t[illegible]size order using the Merge sort algorithm, wh
arrays, sorts [illegible] (u[illegible]erge sort), and then merges the two sub-arrays back

Task 3.3 – Insertion Sort

Generate an array that contains 50 random integers between 1 and 100. It is fine
are repeated.

Write a program to put the array in size order using the Insertion sort algorithm w
the array and inserts it in the right place to its left to ensure that the items checke

Task 3.4 – Quick Sort

Generate an array that cont[illegible]0 r[illegible]dom integers between 1 and 100. It is fine
are repeated.

Write a prog[illegible]o put the array in size order using the Quick sort algorithm whic
middle of the array, then moves the items to the left which are smaller, and to th
then performs a Quick sort on each of the sub-arrays on the left and right.

Task 3.5 – Sort Comparison

Generate an array that contains 5,000 random integers between 1 and 10,000. It
are repeated.

Make five copies of this array. Sort one with a Bubble sort, one with a Merge sort
with a Quick sort, and the last with the inbuilt sort function for the language you
sort takes and display the results, to 5 decimal places.

Note
If your computer takes longer than 30 seconds to run this, or is so fast that it displ
time, then change the size of the array.

Task 3.6 – Linear Search

Generate an array that contains 100 random integers between 1 and 200. It is fi
are repeated.

Generate a single random integer and search the array for it using a Linear search
or not the number is in the array, and which position it is found in.

Task 3.7 – Binary Search

Generate an array that contains 100 random integers between 1 and 200. It is fi
repeated. Sort them in order (using any sort).

Generate a single random integer and search the sorted array for it using a Binary
whether or not the number is in the array, and which position it is found in.

Task 3.8 – Search Comparison

Generate an array that contains 500,000 random integers between 1 and 100,000
numbers are repeated. Sort the array in order.

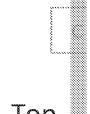
For each number between 1 and 10,000, search for it in the sorted array using eac
Binary search, and the inbuilt search array function. Display to the user how long
10,000 searches.

Why, sometimes, does the Binary search return a slightly different result from the
inbuilt search?

Note
If your computer takes longer than 30 seconds to run this, or is so fast that it displ
taking 0 time, then change the size of the array.

Task 3.9 – Stacks

Create a program that manages a stack where users can ***push*** an item onto a stack or ***pop*** the top item off the stack, displaying the updated stack each time.

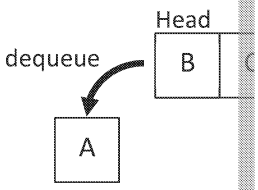


Also give the option to ***peek***, which means to display the current top item on the stack.

The program should continue running until the user opts to exit.

Task 3.10 – Queues

Create a program that manages a stack where users can ***enqueue*** (push) an item onto the tail of a queue or ***dequeue*** (pop) the head item off the queue, displaying the updated queue each time.



Task 3.11 – Trees (harder)

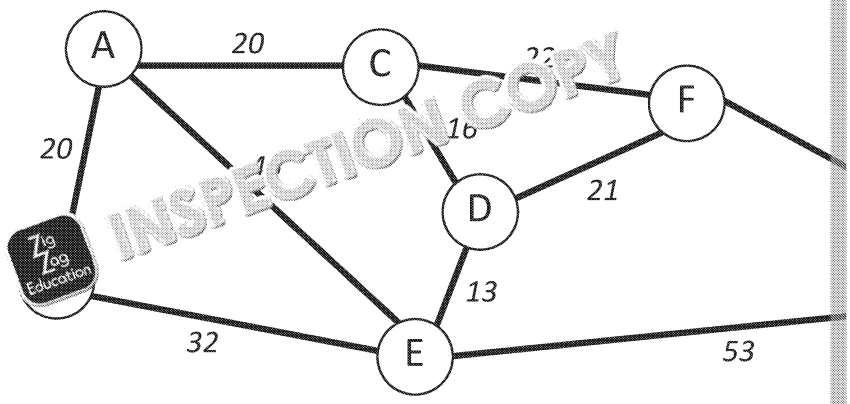
Implement a simple tree structure, where each item (node) in the tree has a name and a root node with ID 1. Allow the user to:

- Add a new node to any other node by giving the name and ID of the parent.
- Remove a node (and all its children if it has any).
- Use an HTML form to provide an interface for interacting with the tree.

Once you have it working, test it by:

- Adding some nodes under the "Root" node.
- Adding child nodes under nodes you have added.
- Removing a node and observing how it affects the tree structure.
- Try adding error handling for invalid parent or node IDs.

Task 3.12 – Graphs; Dijkstra's Shortest P
(harder)



The diagram above shows seven towns (A–G), the main roads between them, and
Write a program that:

1. Stores this information in a *graph* data structure.
2. Outputs a table which shows the distances between all the directly connecte
3. Prompts the user to enter two towns and then uses Dijkstra's Algorithm to fi
   between them.

Section 4 – Games

Now time for a bit of fun! Each of these games is designed to develop your prog
uses complex programming skills, and even the hardest can be completed in few

Your main focus is to get each game to work properly. Al end a small amoun
the presentation. If you are using a web-based pr m ng language such as P
HTML output, then use a style sheet in th le

When you have finished olutions we have provided and compare the
differences b en will help develop your understanding and skills.

Game 1 – Simple Guessing Game

Task

- Generate a random whole number from 1 to 10.
- Ask the user what they think the number is.
- If they get it right tell them "*Well done*", otherwise give them another go, u

Game 2 – Mix Up Fruit and Veg

Task

- Create a function called *getWord()* which cont ay of 50 fruits and vegetables and returns one of them at d
- Randomly shuffle the letters of and display it to the user.
- Ask the user to type in the shuffled word.
- End the e the user gets it right or after they have had three goes.

Game 3 – Hangman

Task

- Create a function called *getRandomWord()* which contains an array of 80 or with between 10 and 15 characters and returns one of them at random. You *ChatGPT*, *Gemini* or *Copilot* to generate the words for you.
- Show an underscore (_) for each letter of the selected word and prompt the a letter.
- If the letter is in the selected word, then display the letter in the right place
- If the letter is not in the selected word, then add it to a displayed list of lette
- If the letter has already been tried, then ignore it.
- The game ends when either the user has ue se the letters in the word, incorrect letters.

Game 4 – Maths Challenge

Task

- Create a function called *generateQuestion()* which generates:
  - two random numbers between 1 and 20
  - either the addition, subtraction, division or multiplication operator, at r
  - the answer (if the division operator is pick[illegible] the division must give whole number)
- Start a timer, then ask the us[illegible] questions as they can answer in 60 s
- Each time they answ[illegible], tell them whether or not they got it right, next qu[illegible].
- After 6[illegible]nds the user is told how many questions they got right (you can question during which the 60 seconds expired).

Game 5 – Quiz

Task

- Create a function called *getQuizQuestions(howManyQuestions)* which contai array of at least 20 questions, each with four multiple-choice answers, and w answer 1, 2, 3 or 4 is correct. The function should return the content of num requested, at random. If more questions are requested than have been crea them all.
- For the game, fetch five questions and then show each question, in order, to
- Once they have answered all five questions, give the[illegible]ir score, and show any of their incorrect answers).

Game [illegible]ock Paper Scissors

Task

- Create a function called *computerGuess()* which returns either "rock", "pape at random.
- Ask the user to pick "rock", "paper" or "scissors".
- The rule is that "rock" beats "scissors", "paper" beats "rock", and "scissors" b It is a draw otherwise. Give one point to the winner, or no points if it is a dra
- Tell the user what they and the computer picked, and who the winner was o
- The first to get to 5 points wins.
- At the end, tell the user who is the winner.

Game 7 – Higher or Lower

Task

- Create a function called *getShuffledCards()* which generates 52 items in an a
  a pack of playing cards, i.e. 2–10, Jack, Queen, King and Ace of each of Hear
  It then shuffles the cards and returns the array.
- Your program should then display the first card f r e user to suggest whet
  or lower. Images for the 52 playing ca re p vided with this resource in t
  names are in the format *02_ f ng*, *ace_of_spaces.png*, etc. Convert t
  filename so you can di the card.
- If the us es r ctly then the next card is displayed, and the game co
- If the n d is the same value (e.g. the 5 of diamonds is worth the same a
  continu
- The game ends when either the user is wrong, or they get all of them right, w
- When the game ends, all the cards should be shown, highlighting the cards t
- If the user gets more than 10 right, tell them that is excellent. If they get mo
  impressive. If they get them all right, tell them that is unbelievable.

*Some languages (e.g. Python) require additional libraries to display images. If you
down and you cannot install libraries, then create a non-graphical version.*

Game 8 – Tic-Tac-Toe

Tic-Tac-Toe is also known as *Noughts and Crosses*.

Task

- Create a 3 × 3 grid of buttons (ne uttons in total).
- The players take turns, ea at the top whose go it is with an X or O:
  - The fi t p ks the square that they want to turn into an X.
  - Ti nd person clicks the square that they want to turn into an O.
  - Thi ocess repeats.
- A player wins if they have three O's or three X's in a row, in a column, or diag
- If all nine squares are filled and no player has won, then it is a draw.

Game 9 – Pairs

Task

- Randomly pick eight card filenames out of the possible 52 playing card image
  The card names are in the format *02_of_clubs.png*, *ace_of_spaces.png*, etc.
- Display 16 cards face down (made up of eight pairs of t cards you have sel
  back of a playing card called *back.png*.
- Indicate at the top whether the curren ye Player 1 or Player 2.
- The current player clicks on t a which are turned over. If the two car
  point and the two ca en away. If they do not match then, upon cli
  are turn ac nd it is the other player's turn.
- Once a ards are turned over, the players are told who won, and how m

Game 10 – Minesweeper

Minesweeper is a classic logic puzzle game where the objective is to clear a minef
mines. The minefield is a grid of blank squares, under some of which are mines. T
to start and it reveals either:

- A number: this number indicates how many mines are adjacent to that squar
- A blank space: if a square has no adjacent mines it will reveal a blank space,
  squares will also be revealed automatically.
- A mine, in which case the game is over.

You win the game by successfully revealing all the squares that do not contain mines.

Task

Create the game of Minesweeper:

- Create a function which:
  - sets up a 10 × 10 array representing 100 squares;
  - randomly allocates 15 squares which contain *exploding mines* (populate
  - populates the rest of the squares with how many mines are next to it (di
- Display the grid to the player. When they click on a square:
  - If the player clicks on a mine it's game over.
  - Otherwise reveal the square they are on. If it is a blank square (i.e. there are no mines next to it) then reveal all the squares around it. If any of them are blank, repeat the process. The player then has another go.
- When the game is over, whether the game is won or lost, display the complete minefield. Display the mines in red. If the player has lost, then display the squares they didn't get that don't contain mines in green.

Hint

When populating th
of mines next to it,
isValidCell(rowNum,
when you check a s
around it, even if yo
those that are off th

Pseudocode Solutions for Sectio

A few key notes on the pseudocode solutions:

- We assume that *print()* will not print a new line, and that *println()* will print a separate functions for printing with and without a new line. Some languages new line, and in HTML a new line is *
*.
- We have assumed that pseudocode can conc[illegible] strings with numbers. [illegible] symbols for concatenation. Python f[illegible] you to either convert numbers to function to insert variables in[illegible]

Section 1 – [illegible]

Task 1.1 – The RANDOM Function

```
println(random(0,10))
println (random(50,100))
println (random(0,1000) / 100)
```

Task 1.2 – Variables

```
random1 = random(0,10)
random2 = random(1000,2000)
println (random1)
println (random2)
println (random1 + random2)
```

Task 1.3 – Concatenation

```
word1 = "super"
word2 = "fast"
word3 = "fly"
result = wo[illegible]" + word2 + " " + word3
println (resu[illegible]
```

Task 1.4 – Arithmetic

```
num1 = random(1,100)
num2 = random(1,10)
println ("First random number from 1 to 100 is " + num1)
println ("Second random number from 1 to 10 is " + num2)
println (num1 + " plus " num2 + " = " + (num1 + num2))
println (num2 + " minus " num1 + " = " + (num2 – num1))
println (num1 + " to the power of " num2 + " = " + (num1 ^ num2))
println (num1 + " + " num2 + " = " + (num1 + num2))
println (num1 + " divided by " num2 + " = " + (num1 / num2))
println (num1 + " divided by " num2 + " = " + [illegible] num2) + " remainder" + ([illegible]
```

Task 1.5 – [illegible]

```
for (i = 1 to [illegible]
    print( i + [illegible])
next
```

Preview of Answers Ends Here