AQA

GCSE

COMPUTER SCIENCE

Paper 1 Computational thinking and programming skills – C#

8525/1A

Diagram Booklet

[Turn over]

FIGURE 1

```
film ← "Godzilla vs. Kong"
year ← 2021
OUTPUT "Please guess a letter"
letter ← USERINPUT
```

FIGURE 2

```
1     num ← USERINPUT
2     IF NOT(num > 1) OR num > 20 THEN
3        OUTPUT "False"
4     ELSEIF num > 1 AND num < 15 THEN
5        OUTPUT "Almost"
6     ELSEIF num MOD 5 = 0 THEN
7        OUTPUT "True"
8     ELSE
9        OUTPUT "Unknown"
10    ENDIF
```

BLANK PAGE

[Turn over]

FIGURE 3

```
1       Random rGen = new Random();
2       int randomNumber;
3
4       Console.WriteLine("Enter a number");
5       int userNumber = Convert.ToInt32(Console.ReadLine());
6       while (userNumber < 1 || userNumber > 100)
7       {
8          Console.WriteLine("Invalid number");
9          userNumber = Convert.ToInt32(Console.ReadLine());
10      }
11      Console.WriteLine("Valid number entered");
12      if (randomNumber == userNumber)
```

```
13      {
14          Console.WriteLine("Number guessed correctly");
15      }
```

[Turn over]

FIGURE 4

```
numberOfGuests ← USERINPUT
numberOfRooms ← USERINPUT
charge ← 25
IF numberOfGuests > 50 THEN
   totalCost ← numberOfGuests * 2
ELSE
   IF numberOfGuests ≥ 25 THEN
      totalCost ← numberOfGuests * 4
   ELSE
      totalCost ← numberOfGuests * 5
   ENDIF
ENDIF
totalCost ← totalCost + (numberOfRooms * 100)
IF totalCost < 1400 THEN
   totalCost ← totalCost + charge
ENDIF
OUTPUT totalCost
```

FIGURE 5

sweetID	sweetName	brand
S1	WINE GUMS	MAYNARDS
S2	COLA CUBES	BERRYMANS
S3	STARBURST	WRIGLEY

[Turn over]

FIGURE 6

```
days ← [10, 15, 4]
sales ← [20, 33, 12]
weeks ← [0, 0, 0]
FOR i ← 0 TO 2
    daysTotal ← days[i] + sales[i]
    weeks[i] ← daysTotal DIV 7
ENDFOR
weeksTotal ← weeks[0] + weeks[1] + weeks[2]
OUTPUT weeksTotal
```

TABLE 2

1	2	author
B1	B2	Book
bookName	i	Real
OUTPUT	String	Boolean

[Turn over]

FIGURE 8

```
static void First(int p1, int p2, int p3)
{
   int v1 = p2 + p3;
   Console.WriteLine(Second(v1, p1));
}

static int Second(int p1, int p2)
{
   int v1 = p1 + p2;
   if (v1 > 12)
   {
      v1 = v1 + Third(p1);
   }
   return v1;
}

static int Third(int p1)
{
   if (p1 > 3)
   {
      return 2;
   }
```

```
    else
    {
        return 0;
    }
}
```

FIGURE 9

USERNAME	PASSWORD
Yusuf5	33kk
Mary80	af5r

[Turn over]

FIGURE 10

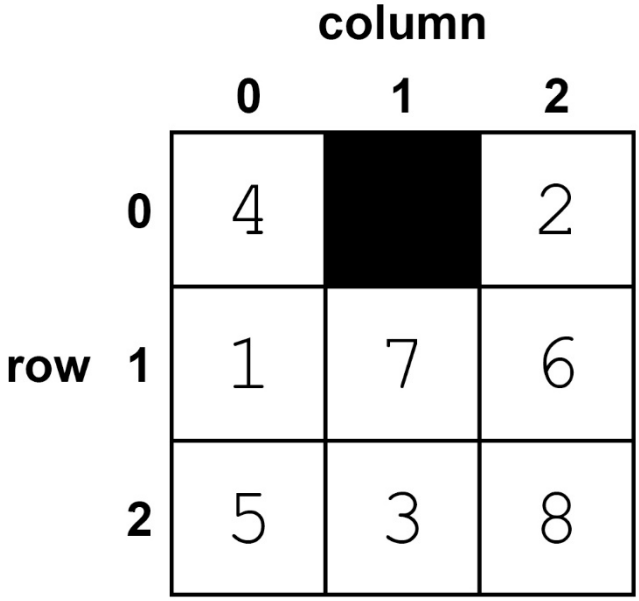
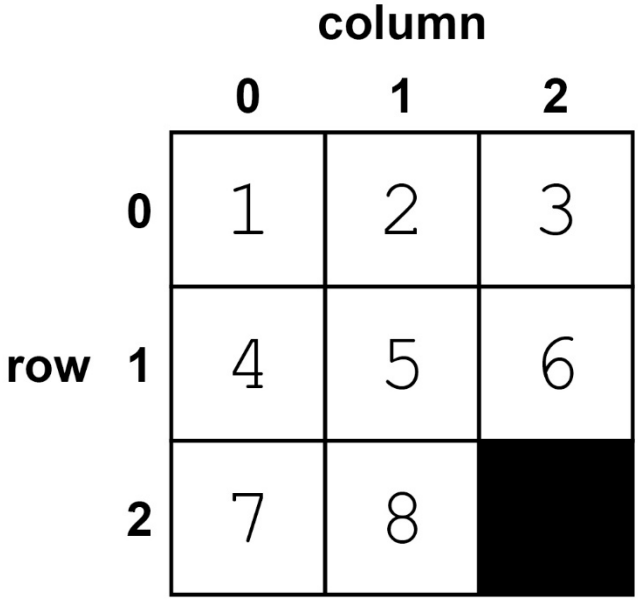


FIGURE 11



BLANK PAGE

[Turn over]

TABLE 3

SUBROUTINE	PURPOSE
`getTile(row, column)`	**Returns the number of the tile on the board in the position** `(row, column)` **For example:** • `getTile(1, 0)` **will return the value** `5` **if it is used on the board in FIGURE 12, provided in the Diagram Booklet** • `getTile(1, 2)` **will return the value** `0` **if it is used on the board in FIGURE 12.**
`move(row, column)`	**Moves the tile in position** `(row, column)` **to the blank space, if the blank space is next to that tile.** **If the position** `(row, column)` **is not next to the blank space, no move will be made.**

	For example: • `move(0, 2)` **would change the board shown in FIGURE 12, provided in the Diagram Booklet, to the board shown in FIGURE 13, provided in the Diagram Booklet** • `move(2, 0)` **would not make a move if used on the board shown in FIGURE 12.**
`displayBoard()`	**Displays the board showing the current position of each tile.**

[Turn over]

FIGURE 12

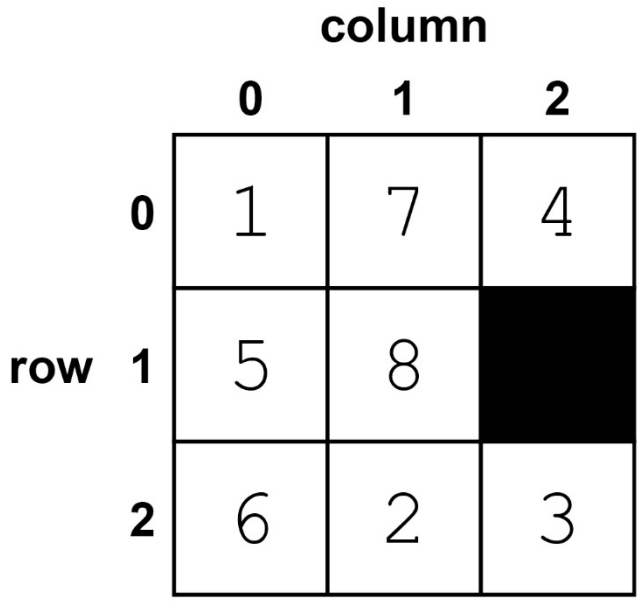
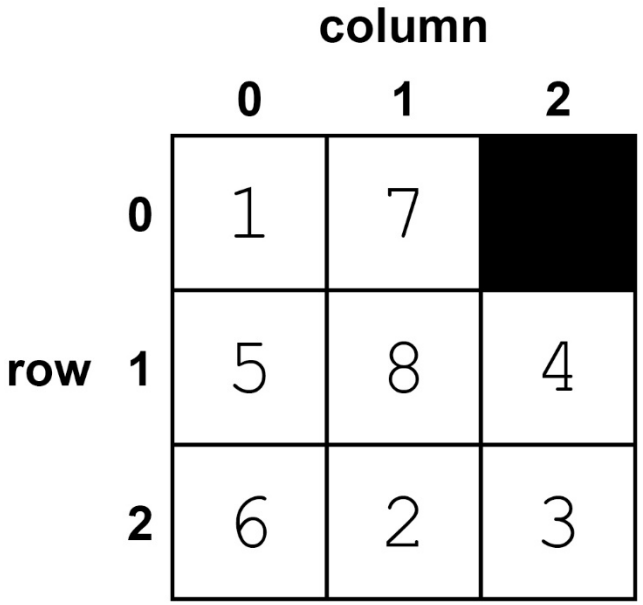


FIGURE 13

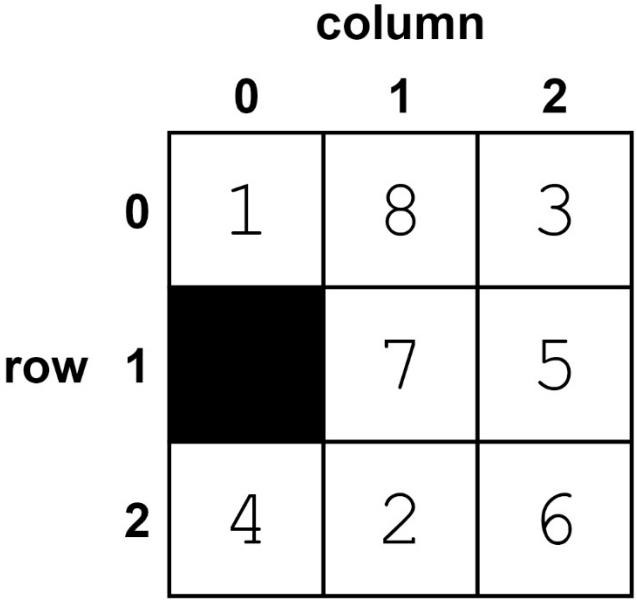


[Turn over]

FIGURE 14

```
if (getTile(1, 0) == 0)
{
   move(2, 0);
}
if (getTile(2, 0) == 0)
{
   move(2, 1);
}
displayBoard();
```

FIGURE 15

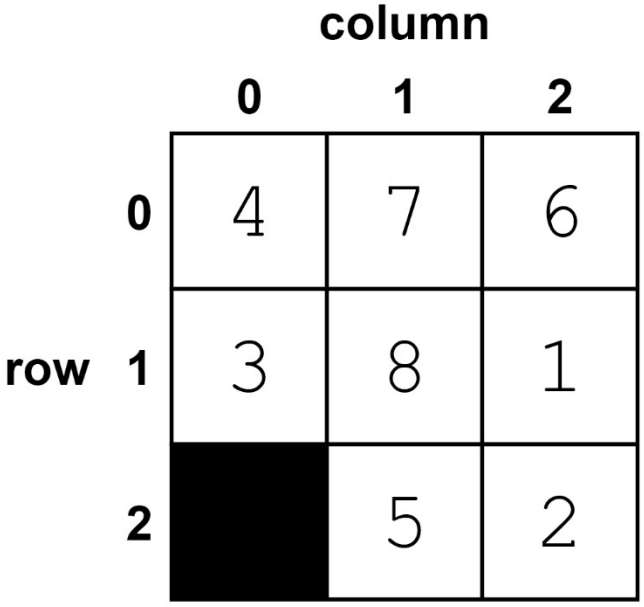


[Turn over]

FIGURE 16

```
int ref1, ref2;
for (int i = 0; i < 3; i++)
{
   for (int j = 0; j < 3; j++)
   {
      if (getTile(i, j) == 0)
      {
         ref1 = i;
         ref2 = j;
      }
   }
}
```

FIGURE 17



[Turn over]

TABLE 4

SUBROUTINE	PURPOSE
`getTile(row, column)`	**Returns the number of the tile on the board in the position** `(row, column)`

FIGURE 18

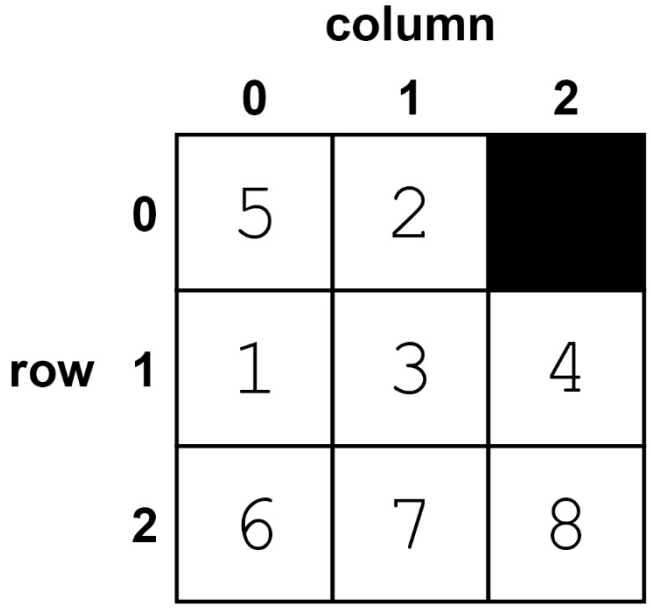
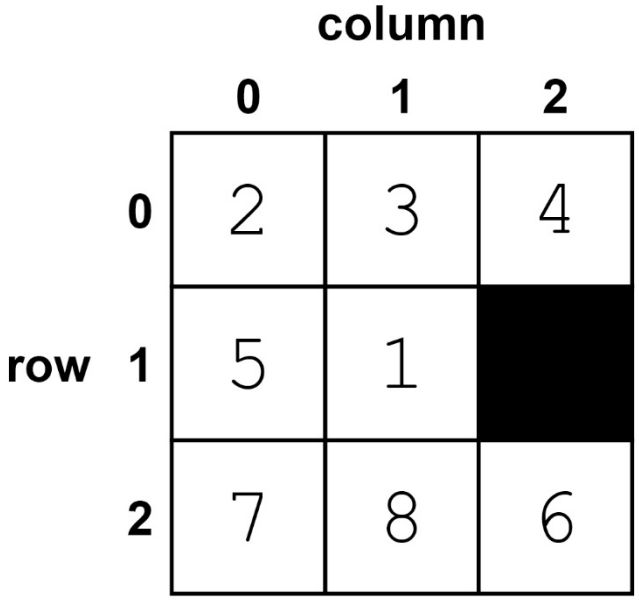


FIGURE 19



[Turn over]

TABLE 5

SUBROUTINE	PURPOSE
`solved()`	**Returns** `true` **if the puzzle has been solved.** **Otherwise returns** `false`
`checkSpace(row, column)`	**Returns** `true` **if there is a blank space next to the tile on the board in the position** `(row, column)` **Otherwise returns** `false`

TABLE 6

SUBROUTINE	PURPOSE
`move(row, column)`	**Moves the tile in position** `(row, column)` **to the blank space, if the blank space is next to that tile.** **If the position** `(row, column)` **is not next to the blank space, no move will be made.**

FIGURE 20

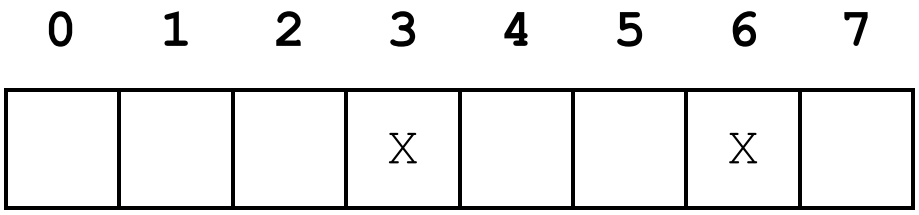


FIGURE 21

- **The player starts at position `0` in a row of cells.**
- **The aim of the game is for the player to reach the end of the row.**
- **At each turn the player must enter either `1` or `2`**
  - **if the player enters `1`, the player's position increases by 1**
  - **if the player enters `2`, the player's position increases by 2**
- **If the player's position goes beyond the end of the row or contains an `X`:**
  - **the message `Bad move` is displayed**
  - **the player goes back to position `0`**
- **These steps are repeated until the player reaches the end of the row.**
- **If the player reaches the end of the row the game is finished.**

FIGURE 22

```
int pos = 0;
int lastPos = row.Length - 1;
while (pos < lastPos)
{
```

END OF DIAGRAM BOOKLET

BLANK PAGE

Copyright information

For confidentiality purposes, all acknowledgements of third-party copyright material are published in a separate booklet. This booklet is published after each live examination series and is available for free download from www.aqa.org.uk.

Permission to reproduce all copyright material has been applied for. In some cases, efforts to contact copyright-holders may have been unsuccessful and AQA will be happy to rectify any omissions of acknowledgements. If you have any queries please contact the Copyright Team.

Copyright © 2024 AQA and its licensors. All rights reserved.