සියලු ම හිමිකම් ඇවිරිණි / All Rights Reserved]

Information and Communication Technology
තොරතුරු හා සන්නිවේදන තාක්ෂණය

2026 QUIZ NO 251 MARKING

2025/08/18
ictfromabc
Ravindu Bandaranayake

1. An action that cannot be taken to prevent plagiarism is,
   රචනා චෞරත්වය වැළැක්වීමට ගත නොහැකි ක්‍රියාමාර්ගයක් වන්නේ,

   1) Using a software to find plagiarism.
      රචනා චෞරත්වය සොයා ගැනීමට මෘදුකාංගයක් භාවිතා කිරීම.
   2) Double citation is one way to avoid plagiarism.
      ද්විත්ව උපුටා දැක්වීම රචනා චෞරත්වය වළක්වා ගැනීමට එක් මාර්ගයකි.
   3) Citation of a quoted part.
      උපුටා ගත් කොටස සඳහන් කිරීම.
   4) Writing what you have to say about the source, instead of the words or ideas of the source.
      මූලාශ්‍රයේ වචන හෝ අදහස් වෙනුවට මූලාශ්‍රය ගැන ඔබට කිව යුතු දේ ලිවීම.
   5) Rewriting some of the words in the source as your own.
      මූලාශ්‍රයේ ඇති සමහර වචන ඔබේම ලෙස නැවත ලිවීම.

   Answer : 5)

   Explanation / පැහැදිලි කිරීම:

   The correct answer to the question "An action that cannot be taken to prevent essay plagiarism is" is indeed option (5) "Rewriting some of the words in the source as your own". This action is not a valid way to prevent plagiarism because it still constitutes as plagiarism, specifically known as paraphrasing plagiarism.
   "රචනා චෞරත්වය වැළැක්වීමට ගත නොහැකි ක්‍රියාමාර්ගයක්" යන ප්‍රශ්නයට නිවැරදි පිළිතුර වන්නේ 5 වන විකල්පයයි. "මූලාශ්‍රයේ ඇති සමහර වචන ඔබේම ලෙස නැවත ලිවීම,". මෙම ක්‍රියාව කොල්ලකෑම වැළැක්වීමට වලංගු ක්‍රමයක් නොවේ, මන්ද එය තවමත් ප්‍රාචීන සොරකම් ("paraphrasing plagiarism") ලෙස විශේෂයෙන් හඳුන්වනු ලබන සොරකම් ලෙසින් සමන්විත වේ.

   Paraphrasing plagiarism occurs when one rephrases the words of a source but does not provide proper citation. Even if the words are changed, the underlying ideas or information are still derived from the original source. Therefore, it's crucial to not only rewrite but also to cite the original source to avoid this type of plagiarism.
   යමෙක් මූලාශ්‍රයක වචන නැවත ප්‍රකාශ කරන නමුත් නිසි උපුටා දැක්වීමක් ලබා නොදෙන විට ප්‍රාචීන සොරකම් (Paraphrasing plagiarism) සිදුවේ. වචන වෙනස් කළද, යටින් පවතින අදහස් හෝ තොරතුරු තවමත් මුල් මූලාශ්‍රයෙන් ලබා ගනී. එමනිසා, මෙම ආකාරයේ කොල්ලකෑම් වළක්වා ගැනීම සඳහා නැවත ලිවීම පමණක් නොව, මුල් මූලාශ්‍රය උපුටා දැක්වීමද ඉතා වැදගත් වේ.

   In contrast, the other options provided such as using plagiarism detection software, double citation, citing quoted parts, and writing your own interpretation of the source are all valid strategies to prevent plagiarism.
   ඊට ප්‍රතිවිරුද්ධව, කොල්ලකෑම් හඳුනාගැනීමේ මෘදුකාංගය (plagiarism detection software) භාවිතා කිරීම, ද්විත්ව උපුටා දැක්වීම, උපුටා ගත් කොටස් උපුටා දැක්වීම සහ මූලාශ්‍රය පිළිබඳ ඔබේම අර්ථකථනය ලිවීම වැනි අනෙකුත් විකල්ප සොරකම් වැළැක්වීම සඳහා වලංගු උපාය මාර්ග වේ.

2. What is the extension of the acronym EEPROM?
EEPROM යන කෙටි යෙදුමෙහි දිගුව කුමක්ද?

1) Electrically Electronically Permanent Read only Memory
2) Electrically Erasable Programmable Read only Memory
3) Electronically Erasable Programmable Read only Memory
4) Electrically Electronically Permanent Random Access Memory
5) Elementary Electrically Programmable Read only Memory

Answer : 2)

Explanation / පැහැදිලි කිරීම:

EEPROM stands for Electrically Erasable Programmable Read-Only Memory. It is a type of non-volatile memory that can be electrically erased and reprogrammed repeatedly. EEPROM is commonly used in various electronic devices for storing small amounts of data that need to be retained even when power is turned off.
EEPROM යනු Electrically Erasable Programmable Read-Only Memory යන්නයි. එය විද්‍යුත් ක්‍රමයෙන් මකා දමා නැවත නැවත ක්‍රමලේඛනය කළ හැකි නශ්‍ය නොවන මතක වර්ගයකි. EEPROM සාමාන්‍යයෙන් විවිධ ඉලෙක්ට්‍රොනික උපාංගවල විදුලිය විසන්ධි වූ විට පවා රඳවා තබා ගත යුතු කුඩා දත්ත ප්‍රමාණයක් ගබඩා කිරීම සඳහා භාවිතා වේ.

1. Non-volatile: EEPROM retains stored data even when power is removed, making it suitable for storing essential data.
නශ්‍ය නොවන: EEPROM බලය ඉවත් කළ විට පවා ගබඩා කළ දත්ත රඳවා තබා ගනී, එය අත්‍යවශ්‍ය දත්ත ගබඩා කිරීම සඳහා සුදුසු වේ.

2. Electrically Erasable: Unlike traditional ROM (Read-Only Memory), EEPROM allows for individual bytes or blocks of data to be erased and rewritten electrically.
විද්‍යුත් වශයෙන් මකා දැමිය හැකි: සම්ප්‍රදායික ROM (Read-Only Memory) මෙන් නොව, EEPROM මඟින් තනි බයිට් හෝ දත්ත කොටස් මැකීමට සහ විද්‍යුත් වශයෙන් නැවත ලිවීමට ඉඩ සලසයි.

3. Programmable: EEPROM can be programmed multiple times, allowing for flexibility in updating stored information as needed.
වැඩසටහන්ගත කළ හැකි: EEPROM කිහිප වතාවක් වැඩසටහන්ගත කළ හැකි අතර, අවශ්‍ය පරිදි ගබඩා කර ඇති තොරතුරු යාවත්කාලීන කිරීමේදී නම්‍යශීලී වීමට ඉඩ සලසයි.

3. $23_{10} + 10010001_2 =$ ……..

Which of the following is the most suitable value to fill in the blank?
හිස්තැන පිරවීම සඳහා වඩාත් සුදුසු අගය කුමක්ද?

1) 10100010_2 2) 167_{10} 3) $\underline{10101000_2}$ 4) 158_{10} 5) 01101000_2

Answer : 3)

Explanation / පැහැදිලි කිරීම:

There are 2 methods to do this,
මෙහි පිළිතුර ලබාගැනීම සඳහා ආකාර දෙකක් ඇත,

Convert 23_{10} into binary(8 bits) and then get the answer.
23_{10} ද්විමය (බිටු 8) බවට පරිවර්තනය කර පිළිතුර ලබා ගන්න.

23_{10}
23 / 2 → 11 - 1 11/2 → 5 - 1 5/2 → 2 - 1 2/2 → 1 - 0 1/2 → 0 - 1
$23_{10} = 00010111_2$
$0\ 0\ 0\ 1\ 0\ 1\ 1\ 1_2$ $+1\ 0\ 0\ 1\ 0\ 0\ 0\ 1_2$ $1\ 0\ 1\ 0\ 1\ 0\ 0\ 0_2$
10101000_2

10101000_2 is the correct answer to fill in the blank and it is among the given answers.
මෙම හිස්තැන පිරවීම සඳහා සුදුසු අගය 10101000_2 වන අතර එය ඉහත සඳහන් පිළිතුරු අතර ඇත.

Convert 10101000_2 into decimal and then get the answer.
10101000_2 **දශමයට පරිවර්තනය කර පිළිතුර ලබා ගන්න.**

10101000_2							
1	0	1	0	1	0	0	0
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	0	32	0	8	0	0	0
128 + 32 + 8 = **168_{10}**							

168_{10} is not among the answers given above.Therefore (3) is the correct answer.
ඉ ඇති පිළිතුරු අතර 168_{10} නොමැත. එමනිසා (3) පිළිතුර පමණක් නිවැරදි වේ.

4. What is the bitwise AND of 11100011_2 and x ?
Here x is the bitwise XOR of 11110000_2 and 10101010_2
11100011_2 **සහ** x **හි බිටු අනුසාරිත** AND **මෙහෙයුම සිදුකිරීමෙන් ලැබෙන පිළිතුර කුමක්ද?**
මෙහි x **යනු** 11110000_2 **සහ** 10101010_2 **හි බිටු අනුසාරිත** XOR **මෙහෙයුමෙහි පිළිතුර වේ.**

1) 11100010_2 2) 10111101_2 3) $\underline{01000010_2}$ 4) 01000000_2 5) 11100011_2

Answer : 3)

Explanation / පැහැදිලි කිරීම:

It has given that x is the bitwise XOR of 11110000_2 and 10101010_2 . So, let's find x.
x **යනු** 11110000_2 **සහ** 10101010_2 **හි බිටු අනුසාරිත** XOR **බව ලබා දී ඇත. එබැවින්** x **සොයමු.**

Truth table for the XOR gate
XOR **ද්වාරය සඳහා සත්‍යතා වගුව**

A	B	F
0	0	0
0	1	1
1	0	1
1	1	0

1	1	1	1	0	0	0	0
1	0	1	0	1	0	1	0
0	1	0	1	1	0	1	0

So, x is 01011010_2. Now let's find the bitwise AND of 11100011_2 and x.
x **යනු** 01011010_2. **දැන්** 11100011_2 **සහ** x **හි බිටු අනුසාරිත** AND **සොයමු.**

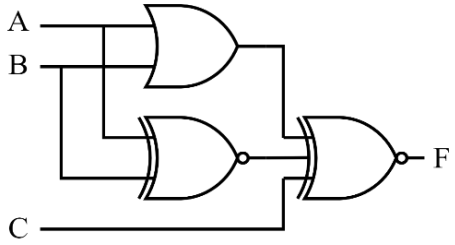
1	1	1	0	0	0	1	1
0	1	0	1	1	0	1	0
0	1	0	0	0	0	1	0

Truth table for the AND gate
AND ද්වාරය සඳහා සත්‍යතා වගුව

A	B	F
0	0	0
0	1	0
1	0	0
1	1	1

Therefore, the answer is 01000010_2.
එබැවින් පිළිතුර 01000010_2 වේ.

5. The values to be given as input to get the output as F=1 are,
F = 1 ලෙස ප්‍රතිදානය ලබාගැනීමට ආදානය ලෙස ලබාදිය යුතු අගයන් වන්නේ,



1) A = 1 , B = 1 , C = 1
2) A = 1 , B = 0 , C = 0
3) A = 0 , B = 0 , C = 0
4) A = 0 , B = 1 , C = 0
5) A = 0 , B = 0 , C = 1

Answer : 5)

Explanation / පැහැදිලි කිරීම:

A	B	C	P= A+B	Q= A XNOR B	F= P XNOR Q XNOR C
1	1	1	1	1	0
1	0	0	1	0	0
0	0	0	0	1	0
0	1	0	1	0	0
0	0	1	0	1	1

Looking at the output as F, only the last one has the output as 1. So the correct answer is number 5.
F ලෙස ඇති ප්‍රතිදානය දෙස බැලූ විට 1 ලෙස ප්‍රතිදානය වී ඇත්තේ අවසාන එකෙහි පමණි. එබැවින් නිවැරදි පිළිතුර වනුයේ අංක 5 ය.

6. Which of the following statements about full adders and half adders is true?
පූර්ණ ආකලකය සහ අර්ධ ආකලකය පිළිබඳ පහත සඳහන් කුමන ප්‍රකාශය සත්‍ය ද?

   1) A half adder can add two binary digits and includes a carry in input.
   අර්ධ ආකලකයට ද්විමය ඉලක්කම් දෙකක් එකතු කළ හැකි අතර carry in ආදානයක් ඇතුළත් වේ.

   2) A full adder can add two binary digits but does not include a carry in input.
   පූර්ණ ආකලකයට ද්විමය ඉලක්කම් දෙකක් එකතු කළ හැකි නමුත් carry in ආදානයක් ඇතුළත් නොවේ.

   3) A half adder produces a sum and a carry out output but cannot handle a carry in input.
   අර්ධ ආකලකය එකතුවක් සහ carry out ප්‍රතිදානයක් නිපදවන නමුත් carry in ආදානයක් හැසිරවිය නොහැක.

   4) A full adder can only add two binary digits and cannot handle a carry in input.
   පූර්ණ ආකලකයට ද්විමය ඉලක්කම් දෙකක් පමණක් එකතු කළ හැකි අතර carry in ආදානයක් හැසිරවිය නොහැක.

   5) Both full adders and half adders require three inputs: two significant bits and a carry in bit.
   පූර්ණ ආකලකය සහ අර්ධ ආකලකය යන දෙකටම ආදාන තුනක් අවශ්‍ය වේ. (සැලකිය යුතු බිටු දෙකක් සහ carry in බිට් එකක්)

   Answer : 3)

   Let's consider each option one by one.
   එක් එක් විකල්පය එකින් එක සලකා බලමු.

   - **Statement / ප්‍රකාශය 1**

   This is incorrect. A half adder adds two binary digits but does not include a carry in input. It only produces a sum and a carry out.
   මෙය නිවැරදි නොවේ. අර්ධ ආකලකයක් ද්විමය ඉලක්කම් දෙකක් එකතු කරන නමුත් ආදානයේ රැගෙන යාමක් ඇතුළත් නොවේ. එය නිෂ්පාදනය කරන්නේ එකතුවක් සහ රැගෙන යාමක් පමණි.

   - **Statement / ප්‍රකාශය**

   This is incorrect. A full adder can add two binary digits and does include a carry in input, which allows it to handle carry-over from previous additions.
   මෙය නිවැරදි නොවේ. පූර්ණ ආකලකයකට ද්විමය ඉලක්කම් දෙකක් එකතු කළ හැකි අතර ආදානයේ රැගෙන යාමක් ඇතුළත් වේ, එය පෙර එකතු කිරීම් වලින් රැගෙන යාම හැසිරවීමට ඉඩ සලසයි.

   - **Statement / ප්‍රකාශය 3**

   This is correct. A half adder takes two binary inputs, produces a sum and a carry out, but does not have a carry in input. It is only suitable for adding two single bits without considering any carry from previous additions.
   මෙය නිවැරදි වේ. අර්ධ ආකලකයක් ද්විමය ආදාන දෙකක් ගනී, එකතුවක් සහ ගෙනයාමක් නිපදවයි, නමුත් ආදානයේ රැගෙන යාමක් නොමැත. එය යෝග්‍ය වන්නේ පෙර එකතු කිරීම් වලින් කිසිදු රැගෙන යාමක් නොසලකා තනි බිටු දෙකක් එකතු කිරීම සඳහා පමණි.

   - **Statement / ප්‍රකාශය 4**

   This is incorrect. A full adder can handle two binary digits and also includes a carry in input, making it capable of handling multi-bit binary addition by taking into account carry-over from previous additions.
   මෙය නිවැරදි නොවේ. පූර්ණ ආකලකයකට ද්විමය ඉලක්කම් දෙකක් හැසිරවිය හැකි අතර ආදානයේ රැගෙන යාමක් ද ඇතුළත් වේ, පෙර එකතු කිරීම් වලින් රැගෙන යාම සැලකිල්ලට ගනිමින් බහු-බිටු ද්විමය එකතු කිරීම හැසිරවීමේ හැකියාව එයට ඇත.

   - **Statement / ප්‍රකාශය 5**

   This is incorrect. While a full adder requires three inputs (two significant bits and a carry in), a half adder only requires two inputs (the two binary digits) and does not have a carry in input.
   මෙය නිවැරදි නොවේ. පූර්ණ ආකලකයකට ආදාන තුනක් (සැලකිය යුතු බිටු දෙකක් සහ රැගෙන යාමක්) අවශ්‍ය වන අතර, අර්ධ ආකලකයකට අවශ්‍ය වන්නේ ආදාන දෙකක් (ද්විමය ඉලක්කම් දෙක) පමණක් වන අතර ආදානය තුළ රැගෙන යාමක් නොමැත.

   So, the correct answer is indeed **3).**
   එබැවින්, නිවැරදි පිළිතුර **3)** වේ.

7. What is the answer when the following Boolean expression is converted to a standard SOP expression?
පහත බූලියානු ප්‍රකාශනය සම්මත SOP **ප්‍රකාශනයක් බවට පරිවර්තනය කළ විට පිළිතුර කුමක්ද?**

$B\overline{A} + \overline{A(CB + CA)}$

1) $\overline{A}BC + \overline{A}B\overline{C} + AB\overline{C} + A\overline{B}\overline{C}$
2) $\underline{\overline{A}BC + \overline{A}B\overline{C} + \overline{A}\overline{B}C + AB\overline{C} + A\overline{B}\overline{C} + \overline{A}\overline{B}\overline{C}}$
3) $ABC + \overline{A}BC + \overline{A}B\overline{C} + \overline{A}\overline{B}C + AB\overline{C} + A\overline{B}\overline{C} + A\overline{B}C + \overline{A}\overline{B}\overline{C}$
4) $ABC + \overline{A}B\overline{C} + \overline{A}\overline{B}C + AB\overline{C} + A\overline{B}C + \overline{A}\overline{B}\overline{C}$
5) $\overline{A}BC + \overline{A}\overline{B}\overline{C}$

Answer : 2)

First, let's trivialize this expression
පළමුව, මෙම ප්‍රකාශනය සුළු කරගනිමු.

$= B\overline{A} + \overline{A(CB + CA)}$	
$= B\overline{A} + \overline{A} + \overline{(CB + CA)}$	De Morgan's law / **ද මුවාර් ප්‍රමේයය**
$= \overline{A} + \overline{(CB + CA)}$	Redundancy law / **සමතිරික්ත නියාය**
$= \overline{A} + \overline{CB}\,\overline{CA}$	De Morgan's law / **ද මුවාර් ප්‍රමේයය**
$= \overline{A} + (\overline{C} + \overline{B})\,(\overline{C} + \overline{A})$	De Morgan's law / **ද මුවාර් ප්‍රමේයය**
$= \overline{A} + \overline{C}\overline{C} + \overline{C}\overline{A} + \overline{B}\overline{C} + \overline{B}\,\overline{A}$	Distributive law/ **විෂටන නියාය**
$= \overline{A} + \overline{C} + \overline{C}\overline{A} + \overline{B}\overline{C} + \overline{B}\overline{A}$	Idempotent law / **තද්ේවභාවී නියාය**
$= \overline{A} + \overline{C} + \overline{B}\overline{C} + \overline{B}\overline{A}$	Redundancy law / **සමතිරික්ත නියාය**
$= \overline{A} + \overline{C} + \overline{B}\overline{A}$	Redundancy law / **සමතිරික්ත නියාය**
$= \overline{A} + \overline{C}$	Redundancy law / **සමතිරික්ත නියාය**

This expression can be converted into a standard SOP expression by simplifying it using Boolean laws as follows.
පහත ආකාරයට බූලියානු නීති භාවිතයෙන් සුළු කිරීමෙන් මෙම ප්‍රකාශනය සම්මත SOP **ප්‍රකාශනයක් බවට පත් කරගත හැකිය.**

$= \overline{A} + \overline{C}$	
$= \overline{A}.1.1 + 1.1.\overline{C}$	Identity law / **සර්වසාම්‍ය නියාය**
$= \overline{A}.(B + \overline{B}).(C + \overline{C}) + (A + \overline{A}).(B + \overline{B}).\overline{C}$	Inverse law / **ප්‍රතිලෝම නියාය**
$= (\overline{A}B + \overline{A}\overline{B})(C + \overline{C}) + (AB + A\overline{B} + \overline{A}B + \overline{A}\overline{B}).\overline{C}$	Distributive law / **විෂටන නියාය**
$= \overline{A}BC + \overline{A}B\overline{C} + \overline{A}\overline{B}C + \overline{A}\overline{B}\overline{C} + AB\overline{C} + A\overline{B}\overline{C} + \overline{A}B\overline{C} + \overline{A}\overline{B}\overline{C}$	Distributive law / **විෂටන නියාය**
$= \overline{A}BC + \overline{A}B\overline{C} + \overline{A}\overline{B}C + \overline{A}\overline{B}\overline{C} + AB\overline{C} + A\overline{B}\overline{C}$	Idempotent law / **තද්ේවභාවී නියාය**

So, the correct answer number is 2.
එමනිසා, නිවැරදි පිළිතුරු අංකය 2 **වේ.**

8. What dialogue does Dominic say in the following Python code?
පහත පයිතන් කේතයට අනුව Dominic පවසන සංවාද කොටස කුමක්ද?

```
character = "Dom"
dialogue = "Dom"

if character == "Dom":
    if dialogue == "Dom":
        print("Dominic: I don't have friends. I got family.")
    else:
        print("O'Conner: That's why we're brothers, because you did too.")
else:
    print("But remember, it's all about family.")
```

1) "I live my life a quarter mile at a time."
2) "I don't have friends. I got family."
3) "That's why we're brothers, because you did too."
4) "You don't turn your back on family."
5) Dominic: I don't have friends. I got family.

Answer : 5)

Explanation / පැහැදිලි කිරීම:

In the provided Python code, the variables `character` and `dialogue` are both set to the string `"Dom"`. The code then checks whether the value of `character` is equal to `"Dom"`, which is true in this case. Following this, it checks whether the value of `dialogue` is also `"Dom"`, which is again true. Because both conditions are satisfied, the code executes the `print` statement within the nested `if` block, outputting the line: **`"Dominic: I don't have friends. I got family."`**
සපයා ඇති පයිතන් කේතයෙහි, විචල්‍ය `character` සහ `dialogue` යන දෙකම `"Dom"` string එකට සකසා ඇත. කේතය පසුව `character` අගය `"Dom"` ට සමාන දැයි පරීක්ෂා කරයි, එය මෙම නඩුවේ සත්‍ය වේ. මෙය අනුගමනය කරමින්, `dialogue` අගය `"Dom"` ද යන්න පරීක්ෂා කරයි, එය නැවතත් සත්‍ය වේ. කොන්දේසි දෙකම තෘප්තිමත් වන නිසා, කේතය nested `if` block එක තුළ `print` ප්‍රකාශය ක්‍රියාත්මක කර, රේඛාව ප්‍රතිදානය කරයි: **`"Dominic: I don't have friends. I got family."`**

Therefore, the correct answer is 5).
එබැවින්, නිවැරදි පිළිතුර 5) වේ.

9. What will be the output of the following Python code?
පහත පයිතන් කේතයේ ප්‍රතිදානය කුමක් වේද?

```
numbers = [1, 2, 3, 4, 5]
squares = [n**2 for n in numbers if n % 2 == 0]
print(squares)
```

1) [1, 4, 9, 16, 25]
2) [1, 4, 9, 16]
3) [4, 16]
4) [1, 9, 25]
5) [4, 9, 16, 25]

Answer : 3)

Explanation / පැහැදිලි කිරීම:

`numbers = [1, 2, 3, 4, 5]`
This line creates a list called `numbers` containing the integers `[1, 2, 3, 4, 5]`.
මෙම රේඛාව නිබිල `[1, 2, 3, 4, 5]` අඩංගු `numbers` නම් ලැයිස්තුවක් නිර්මාණය කරයි.

`squares = [n**2 for n in numbers if n % 2 == 0]`
This is a list comprehension that creates a new list called `squares`.
මෙය `squares` නම් අලුත් ලයිස්තුවක් නිර්මාණය කරන list comprehension එකකි.

`for` loop iterates over each element `n` in the `numbers` list. `if n % 2 == 0` checks whether the number n is even (when divided by 2, it leaves a remainder of 0). For each even number, it calculates the square of n (`n**2`) and includes it in the `squares` list.
`for` ලූපය සඳහා `numbers` ලැයිස්තුවේ එක් එක් මූලාංග `n` මත පුනරාවර්තනය වේ. `if n % 2 == 0` අංකය ඉරට්ටේ දැයි පරීක්ෂා කරයි (2 න් බෙදූ විට, එයට `0` ක් ඉතිරි වේ). එක් එක් ඉරට්ටේ සංඛ්‍යාව සඳහා, එය `n` හි වර්ගය (`n**2`) ගණනය කර එය `squares` ලැයිස්තුවට ඇතුළත් කරයි.

`print(squares)`
This line prints the list `squares`.
මෙම රේඛාව `squares` ලැයිස්තුව මුද්‍රණය කරයි.

From the `numbers` list `[1, 2, 3, 4, 5]`, the even numbers are 2 and 4.
`[1, 2, 3, 4, 5]` ඇතුළත් `numbers` ලැයිස්තුවෙන් ඉරට්ටේ අංක 2 සහ 4 වේ.

- The square of 2 is 4 (`2**2 = 4`).
  2 හි වර්ගය 4 වේ. (`2**2 = 4`)
- The square of 4 is 16 (`4**2 = 16`).
  4 හි වර්ගය 16 වේ. (`4**2 = 16`)

Therefore, the output is 3) `[4, 16]`.
එබැවින්, ප්‍රතිදානය 3) `[4, 16]` වේ.

10. What is the output of the following Python code?
පහත පයිතන් කේතයේ ප්‍රතිදානය කුමක් ද?

```
def special_algorithm(arr):
    n = len(arr)
    for i in range(n):
        min_idx = i
        for j in range(i+1, n):
            if arr[j] < arr[min_idx]:
                min_idx = j
        arr[i], arr[min_idx] = arr[min_idx], arr[i]
    return arr
arr = [64, 25, 12, 22, 11]
finalized_arr = special_algorithm(arr)
print(finalized_arr)
```

1) `[64, 25, 12, 22, 11]`
2) `[11, 12, 22, 25, 64]`
3) `[11, 22, 12, 25, 64]`
4) `[25, 64, 22, 12, 11]`
5) `[11, 22, 25, 64, 12]`

Answer : 2)

Explanation / පැහැදිලි කිරීම:

Above is the selection sort algorithm. You can explain what happens there by the given image.
ඉහතින් දැක්වෙන්නේ Selection sort algorithm එක වේ. එහිදී සිදුවන සිදුවීම ඔබට දී ඇති රූපය මගින් පැහැදිලි කරගත හැකිය.

Also, you don't need to know about this algorithm in advance to write answers for this question. This can be a constructive question. It may be an advantage for you to know this.
එමෙන්ම මෙහිදී මෙම ප්‍රශ්නය සඳහා පිළිතුරු ලිවීමට ඔබ මෙම ඇල්ගොරිතමය පිළිබඳව පෙර සිට දැන සිටිය යුතු නැත. මෙය නිර්මාණාත්මක ප්‍රශ්නයක් විය හැක. ඔබ මෙය දැන සිටීම වාසියක් විය හැක.

If so, everything that happens here can be explained by the following table.
එසේ නම් මෙහිදී සිදුවන සියලු දෑ පහත වගුව මගින් පැහැදිලි කරගත හැකිය.

Iteration	i	j	min_idx	arr
Initial				[64, 25, 12, 22, 11]
1	0	1	1	[25, 64, 12, 22, 11]
1	0	2	2	[12, 64, 25, 22, 11]
1	0	3	2	[12, 64, 25, 22, 11]
1	0	4	4	[11, 64, 25, 22, 12]
2	1	2	2	[11, 25, 64, 22, 12]
2	1	3	3	[11, 22, 64, 25, 12]
2	1	4	4	[11, 12, 64, 25, 22]
3	2	3	3	[11, 12, 25, 64, 22]
3	2	4	4	[11, 12, 22, 64, 25]
4	3	4	4	[11, 12, 22, 25, 64]

According to this table the answer is `[11, 12, 22, 25, 64]`.
මෙම වගුවට අනුව පිළිතුර වන්නේ `[11, 12, 22, 25, 64]` වේ.