සියලු ම හිමිකම් ඇවිරිණි /All Rights Reserved]

Ravindu Bandaranayake - ictfromabc Ravindu Bandaranayake - ictfromabc Ravindu Bandaranayake - ictfromabc

Information and Communication Technology
තොරතුරු හා සන්නිවේදන තාක්ෂණය

2026 QUIZ NO 385 MARKING

2025/12/30
ictfromabc
Ravindu Bandaranayake

1. Computer technology has advanced significantly, leading to the development of more compact and efficient devices. Which of the following innovation played a crucial role in increasing the speed of computers?
පරිගණක තාක්ෂණය සැලකිය යුතු ලෙස දියුණු වී ඇති අතර එය වඩාත් සංයුක්ත හා කාර්යක්ෂම උපාංග සංවර්ධනය කිරීමට හේතු වේ. පහත සඳහන් කුමන නවෝත්පාදනය පරිගණකයේ වේගය වැඩි කිරීමට තීරණාත්මක කාර්යභාරයක් ඉටු කළේද?

1) Magnetic Drum. චුම්බක බෙර. 2) Transistors. ට්‍රාන්සිස්ටර. 3) Magnetic Tape චුම්බක පටි. 4) Punched Cards. සිදුරුපත්. 5) Mechanical Relays. යාන්ත්‍රික රිලේ.

Answer : 2)

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

- **Magnetic Drum / චුම්භක භර**
  This was an early form of computer memory, but it didn't significantly increase the speed of computers. It was slow compared to modern storage systems.
  මෙය පරිගණක මතකයේ මුල් ආකාරයකි, නමුත් එය පරිගණකවල වේගය සැලකිය යුතු ලෙස වැඩි කළේ නැත. නවීන ගබඩා පද්ධති හා සසඳන විට එය මන්දගාමී විය.

- **Transistors / ට්‍රාන්සිස්ටර**
  Transistors replaced vacuum tubes and allowed computers to become faster, more reliable, and smaller. This was a crucial innovation for improving speed.
  ට්‍රාන්සිස්ටර රික්තක නල ප්‍රතිස්ථාපනය කර පරිගණක වේගවත්, වඩා විශ්වාසදායක සහ කුඩා වීමට ඉඩ සලසයි. මෙය වේගය වැඩි දියුණු කිරීම සඳහා තීරණාත්මක නවෝත්පාදනයක් විය.

- **Magnetic Tape / චුම්බක පටි**
  This was used for data storage but didn't directly improve the speed of computers.
  මෙය දත්ත ගබඩා කිරීම සඳහා භාවිතා කරන ලද නමුත් පරිගණකවල වේගය කෙලින්ම වැඩි දියුණු කළේ නැත.

- **Punched Cards / සිදුරුපත්**
  These were used in early computers for input, but they didn't enhance the speed.
  මේවා මුල් පරිගණකවල ආදානය සඳහා භාවිතා කරන ලද නමුත් ඒවායේ වේගය වැඩි කළේ නැත.

- **Mechanical Relays / යාන්ත්‍රික රිලේස්**
  Relays were used in early computers to switch circuits, but they were slow compared to later technologies.
  පරිපථ මාරු කිරීම සඳහා මුල් පරිගණකවල රිලේ භාවිතා කළ නමුත් පසුකාලීන තාක්ෂණයන්ට සාපේක්ෂව ඒවා මන්දගාමී විය.

According to the above statements, we can clearly see that transistors has advanced significantly, leading to the development of more compact and efficient devices.
ඉහත ප්‍රකාශයන්ට අනුව, ට්‍රාන්සිස්ටර සැලකිය යුතු ලෙස දියුණු වී ඇති බව අපට පැහැදිලිව දැකගත හැකිය, එය වඩාත් සංයුක්ත හා කාර්යක්ෂම උපාංග සංවර්ධනය කිරීමට හේතු වේ.

So, the correct answer is 2).
එබැවින්, නිවැරදි පිළිතුර 2) වේ.

2. Which of the following numbers has the highest value?
පහත දක්වා ඇති සංඛ්‍යා වලින් වැඩිම වටිනාකමක් සහිත සංඛ්‍යාව කුමක්ද?

1) 101 2) 57_8 3) 93_{16} 4) 131_8 5) 1101001_2

Answer : 3)

First all the numbers should be converted into decimal numbers.
පළමුව සංඛ්‍යා සියල්ල දශමය සංඛ්‍යා බවට පරිවර්තනය කර ගත යුතුය.

57_8 = ...$_{10}$	
5	7
8^1	8^0
8*5	1*7
40	7
$57_8 = 47_{10}$	

93_{16} = ...$_{10}$	
9	3
16^1	16^0
16*9	1*3
144	3
$93_{16} = 147_{10}$	

131_8 = ...$_{10}$		
1	3	1
8^2	8^1	8^0
64*1	8*3	1*1
64+24+1 = 89		
$131_8 = 89_{10}$		

1101001_2 = ...$_{10}$						
1	1	0	1	0	0	1
2^6	2^5	2^4	2^3	2^2	2^1	2^0
64*1	32*1	16*0	8*1	4*0	2*0	1*1
64+32+8+1 = 105						
$1101001_2 = 105_{10}$						

1) 101 2) $57_8 = 47_{10}$ 3) $93_{16} = 147_{10}$ 4) $131_8 = 89_{10}$ 5) $1101001_2 = 105_{10}$

Then 93_{16} can be taken as the highest value.
ඉන්පසු එයින් වැඩිම වටිනාකමක් සහිත අගය ලෙස 93_{16} ගත හැක.

3. Which of the following statements correctly describes the difference between a half adder and a full adder in digital circuits? අංකිත පරිපථවල අර්ධ ආකලකයක් සහ පූර්ණ ආකලකයක් අතර වෙනස නිවැරදිව විස්තර කරන්නේ පහත සඳහන් කුමන ප්‍රකාශයද?

1) A half adder can add two binary numbers and a carry input, while a full adder can only add two binary numbers. අර්ධ ආකලකයකට ද්විමය අංක දෙකක් සහ carry ආදානයක් එකතු කළ හැකි අතර පූර්ණ ආකලකයකට එකතු කළ හැක්කේ ද්විමය අංක දෙකක් පමණි.
2) A half adder can add two binary numbers, while a full adder can add four binary numbers. අර්ධ ආකලකයකට ද්විමය සංඛ්‍යා දෙකක් එකතු කළ හැකි අතර පූර්ණ ආකලකයකට ද්විමය සංඛ්‍යා හතරක් එකතු කළ හැකිය.
3) A half adder can add two binary numbers, while a full adder can add two binary numbers in lower duration. අර්ධ ආකලකයකට ද්විමය සංඛ්‍යා දෙකක් එකතු කළ හැකි අතර පූර්ණ ආකලකයකට අඩු කාලයකදී ද්විමය සංඛ්‍යා දෙකක් එකතු කළ හැක.
4) A half adder can add two binary numbers, while a full adder can add two binary numbers with greater accuracy. අර්ධ ආකලකයකට ද්විමය සංඛ්‍යා දෙකක් එකතු කළ හැකි අතර පූර්ණ ආකලකයකට වැඩි නිරවද්‍යතාවයකින් ද්විමය සංඛ්‍යා දෙකක් එකතු කළ හැක.
5) A half adder can add two binary numbers, while a full adder can add two binary numbers and handle overflow conditions. අර්ධ ආකලකයකට ද්විමය සංඛ්‍යා දෙකක් එකතු කළ හැකි අතර පූර්ණ ආකලකයකට ද්විමය සංඛ්‍යා දෙකක් එකතු කර overflow තත්ත්වයන් හැසිරවිය හැක.

Answer : 5)

The correct answer is 5). A half adder can add two binary numbers (specifically, it adds two bits) but does not account for any carry input from previous additions. It only produces a sum and a carry output. On the other hand, a full adder can add two binary numbers along with a carry input from a previous operation. This allows the full adder to handle overflow conditions, where a carry might be generated from a lower bit addition and needs to be added to the next bit. Therefore, a full adder is more capable as it can manage both the addition of binary numbers and overflow conditions, making it suitable for multi-bit binary addition.

නිවැරදි පිළිතුර 5) වේ. අර්ධ ආකලකයකට ද්විමය සංඛ්‍යා දෙකක් එකතු කළ හැකිය (විශේෂයෙන්, එය බිටු දෙකක් එකතු කරයි) නමුත් පෙර එකතු කිරීම් වලින් කිසිදු රැගෙන ආ ආදානයක් සඳහා ගණන් නොගනී. එය නිෂ්පාදනය කරන්නේ එකතුවක් සහ ගෙනයාමක් පමණී. අනෙක් අතට, පූර්ණ ආකලකයකට පෙර මෙහෙයුමකින් ගෙන ආ ආදානයක් සමඟ ද්විමය අංක දෙකක් එකතු කළ හැකිය. මෙය පූර්ණ ආකලකයකට overflow තත්වය හැසිරවීමට ඉඩ සලසයි, එහිදී අඩු බිට් එකතු කිරීමකින් රැගෙන යාමක් ජනනය විය හැකි අතර ඊළඟ බිට් එකට එකතු කිරීමට අවශ්‍ය වේ. එබැවින්, බහු-බිට් ද්විමය එකතු කිරීම සඳහා සුදුසු වන පරිදි ද්විමය සංඛ්‍යා එකතු කිරීම සහ overflow තත්වය දෙකම කළමනාකරණය කළ හැකි බැවින් පූර්ණ ආකලකයකට වඩා හැකියාව ඇත.

So, the correct answer is 5).
ඒබැවින්, නිවැරදි පිළිතුර 5) වේ.

4. Consider the relations below.
පහත දැක්වෙන සම්බන්ධතා සලකා බලන්න.

```
Employee(Empid,EmpName,Address,Salary,DepID)
```

Which answer correctly represents the SQL command to increase the salary of all employees whose `DepID = "D101"` by 5000.
`DepID = "D101"` වන සියලුම සේවකයින්ගේ වැටුප 5000 කින් වැඩි කිරීම සඳහා වන SQL විධානය නිවැරැදිව දැක්වෙන පිළිතුර කුමක්ද?

1) `UPDATE Employee ADD Salary=Salary+5000 WHERE DepID="D101";`
2) `UPDATE INTO Employee SET Salary=Salary+5000 WHERE DepID="D101";`
3) `UPDATE Employee SET Salary=Salary+5000;`
4) `UPDATE SET Salary=Salary+5000 From Employee WHERE DepID="D101";`
5) `UPDATE Employee SET Salary=Salary+5000 WHERE DepID="D101";`

Answer: 5)

The UPDATE command in SQL is as follows:
SQL හි UPDATE විධානය පහත පරිදි වේ:

```
UPDATE tablename
SET column1name = value1, column2name = value2
WHERE condition;
```

Therefore, we can correctly represent the SQL command to increase the salary of all employees whose DepID = "D101" by 5000 as follows:
එබැවින්, DepID = “D101” හි වැටුප 5000 කින් වැඩි කිරීම සඳහා SQL විධානය පහත පරිදි නිවැරදිව නිරූපණය කළ හැකිය:

```
UPDATE Employee
SET Salary=Salary+5000
WHERE DepID="D101";
```

So, the correct answer is 5.
එබැවින් නිවැරදි පිළිතුර 5 වේ.

If we don’t specify “`WHERE`”, like in answer 5; SQL will increase the salary of all employees by 5000.
පිළිතුර 5 මෙන් “`WHERE`” යන්න සඳහන් නොකරන්නේ නම්; SQL, සියලුම සේවකයින්ගේ වැටුප 5000 කින් වැඩි කරයි.

5. Consider the table Products with the column PCode. Which query retrieves all products whose code contains exactly 5 characters, ending with the letter "Z"?
PCode නමින් තීරුවක් සහිත නිෂ්පාදන (Products) වගුව සලකා බලන්න. "Z" අකුරෙන් අවසන් වන කේතයේ හරියටම අක්ෂර 5ක් අඩංගු සියලුම නිෂ්පාදන ලබා ගන්නේ කුමන විමසුමද?

1) `SELECT * FROM Products WHERE PCode LIKE 'Z____';`
2) `SELECT * FROM Products WHERE PCode LIKE '____Z';`
3) `SELECT * FROM Products WHERE PCode LIKE '%Z____';`
4) `SELECT * FROM Products WHERE PCode LIKE '____Z%';`
5) `SELECT * FROM Products WHERE PCode LIKE '%Z';`

Answer: 2)

Explanation of each option / එක් එක් විකල්පය පැහැදිලි කිරීම:

Option / විකල්ප 1)
This query retrieves products whose code starts with "Z" and is followed by exactly four characters.
මෙම විමසුම "Z" වලින් ආරම්භ වන කේතය සහ හරියටම අනුලකුණු හතරකින් අනුගමනය කරන නිෂ්පාදන ලබා ගනී.

Option / විකල්ප 2)
This query retrieves products whose code ends with "Z" and has exactly four characters before it (total 5 characters). The underscore (_) wildcard matches exactly one character, so "____Z" ensures there are exactly four characters before "Z". This is the correct answer.
මෙම විමසුම මඟින් කේතය "Z" වලින් අවසන් වන සහ ඊට පෙර හරියටම අක්ෂර හතරක් ඇති නිෂ්පාදන ලබා ගනී (මුළු අක්ෂර 5) යටි ඉරි (_) wildcard එක අක්ෂරයකට හරියටම ගැලපේ, එබැවින් "____Z" "Z" ට පෙර හරියටම අක්ෂර හතරක් ඇති බව සහතික කරයි. මෙය නිවැරදි පිළිතුරයි.

Option / විකල්ප 3)
This query retrieves products containing any number of characters (0 or more) before "Z" and exactly four characters after "Z".
මෙම විමසුම "Z" ට පෙර ඕනෑම අක්ෂර ගණනක් (0 හෝ ඊට වැඩි) සහ "Z" ට පසුව හරියටම අක්ෂර හතරක් අඩංගු නිෂ්පාදන ලබා ගනී.

Option / විකල්ප 4)
This query retrieves products with exactly four characters before "Z" but allows for any number of characters (0 or more) after "Z".
මෙම විමසුම "Z" ට පෙර හරියටම අක්ෂර හතරක් සහිත නිෂ්පාදන ලබා ගන්නා නමුත් "Z" ට පසුව ඕනෑම අක්ෂර ගණනකට (0 හෝ ඊට වැඩි) ඉඩ ලබා දේ.

Option / විකල්ප 5)
This query retrieves products whose code ends with "Z", but it does not restrict the length of the code or the number of characters before "Z".
මෙම විමසුම "Z" සමඟ කේතය අවසන් වන නිෂ්පාදන ලබා ගනී, නමුත් එය කේතයේ දිග හෝ "Z" ට පෙර අක්ෂර ගණන සීමා නොකරයි.

Therefore, the correct answer is 2).
එබැවින්, නිවැරදි පිළිතුර 2) වේ.

6. What is eliminated when a table goes from 1NF to 2NF ?
වගුවක් 1NF සිට 2NF දක්වා යනවිට ඉවත් වන්නේ කුමක්ද?

1) Candidate keys / අපේක්ෂක යතුර
2) Transitive dependencies / සංක්‍රාන්තික පරායක්තතාව
3) Partial dependencies / ආංශික පරායක්තතාව
4) Repeating groups / පුනරාවර්තන කාණ්ඩ
5) Nove of the above / ඉහත කිසිවක් නොවේ.

Answer: 3)

Explanation / පැහැදිලි කිරීම:

A table is in 2NF (Second Normal Form) if it satisfies the following conditions:
පහත සඳහන් කොන්දේසි සපුරාලන්නේ නම් වගුවක් 2NF (දෙවන ප්‍රමත අවස්ථාව) වේ:

- It is in 1NF (First Normal Form) / එය 1NF (පළමු ප්‍රමත අවස්ථාව) හි තිබීම:

  This means that the table should have only atomic (indivisible) values, and there should be no repeating groups or arrays within a single column.
  මෙයින් අදහස් කරන්නේ වගුවේ පරමාණුක (බෙදා දිය නොහැකි) අගයන් පමණක් තිබිය යුතු අතර තනි තීරුවක් තුළ පුනරාවර්තන කණ්ඩායම් හෝ අරා නොතිබිය යුතු බවයි.

- It does not have any Partial Dependencies / එයට කිසිදු ආංශික පරයක්තතා නොමැත:

  This means that non-key attributes should depend on the whole candidate key, not just a part of it.
  මෙයින් අදහස් කරන්නේ යතුර නොවන උපලක්ෂණ එහි කොටසක් පමණක් නොව සම්පූර්ණ අපේක්ෂක යතුර මත රඳා පවතින බවයි.

Therefore, the correct answer is 3).
එබැවින්, නිවැරදි පිළිතුර 3) වේ.

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

```
def reverse_add(n):
    rev_num = int(str(n)[::-1])
    return n + rev_num

num = 500
while num < 1000:
    num = reverse_add(num)
    print(num, end=" ")
```

1) 505 1110   2) 555 1010   3) 505 1000   4) 505 1010   5) 500 1010

Answer : 4)

The code defines a function `reverse_add(n)` that reverses the digits of `n` and adds the reversed number to `n`. The initial value of `num` is `500`, and the `while` loop runs as long as `num` is less than `1000`. In the first iteration, `reverse_add(500)` reverses `500` to get 5, and adds it to `500`, resulting in 505. In the next iteration, `reverse_add(505)` reverses `505` to get `505`, and adds it to `505`, resulting in `1010`. Since `1010` is greater than `1000`, the loop stops. The output is `505 1010`.
කේතය `n` හි ඉලක්කම් ආපසු හරවා `n` වෙත ප්‍රතිලෝම අංකය එකතු කරන `reverse_add(n)` ශ්‍රිතයක් අර්ථ දක්වයි. `num` හි ආරම්භක අගය `500` වන අතර, `while` ලූපය `num` `1000` ට වඩා අඩු වන තාක් ධාවනය වේ. පළමු පුනරාවර්තනයේදී, `reverse_add(500)` `5` ලබා ගැනීමට `500` ආපසු හරවා, `500` ට එකතු කරයි., ප්‍රතිඵලය `505` වේ. මීළඟ පුනරාවර්තනයේදී, `reverse_add(505))` `505` ලබා ගැනීමට `505` ප්‍රතිලෝම කර, `505` ට එකතු කරයි, ප්‍රතිඵලයක් ලෙස `1010` ලබාදේ. `1010` `1000` ට වඩා වැඩි බැවින්, ලූපය නතර වේ. ප්‍රතිදානය `505 1010` වේ.

8. What will be the output of the following python code?
පහත පයිතන් කේතයේ ප්‍රතිදානය කුමක් වේවිද?

```
def add_numbers(a, b=5):
    return a + b

result = add_numbers(10)
print(result)
```

1) 10   2) 5   3) 0   4) 15   5) ERROR

Answer : 4)

Explanation / පැහැදිලි කිරීම

In this code, a function add_numbers is defined with two parameters a and b.
මෙම කේතයෙහි, add_numbers ශ්‍රිතයක් a සහ b යන පරාමිති දෙකකින් අර්ථ දක්වා ඇත.

The parameter b is assigned a default value of 5 . This means that if no second argument is provided when the function is called, b will automatically be set to 5.
b පරාමිතියට 5 හි පෙරනිමි අගයක් පවරනු ලැබේ. මෙයින් අදහස් කරන්නේ ශ්‍රිතය කැඳවන විට දෙවන තර්කයක් ලබා නොදෙන්නේ නම්, b ස්වයංක්‍රීයව 5 ලෙස සකසනු ඇති බවයි.

Inside the function, the sum of a and b is returned. The function is then called with the argument 10 for a, while no argument is provided for b, so it defaults to 5.
ශ්‍රිතය තුළ, a සහ b හි එකතුව ආපසු ලබා දෙනු ලැබේ. එවිට ශ්‍රිතය a සඳහා තර්කය 10 සමඟින් හඳුන්වනු ලබන අතර, b සඳහා තර්කයක් සපයා නැත, එබැවින් එය 5 වෙත පෙරනිමි වේ.

As a result, the function adds 10 and 5, returning 15, which is then printed.
ප්‍රතිඵලයක් වශයෙන්, ශ්‍රිතය 10 සහ 5 එකතු කරයි, 15 ආපසු ලබා දෙයි, පසුව මුද්‍රණය කරනු ලැබේ

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

```
data = {'a': 10, 'b': 20, 'c': 30}
sum = 0

for key in data:
    sum += data[key]
    if sum > 50:
        break

print(sum)
```

1) 60   2) 59   3) 61   4) 65   5) None of the above.

Answers : 1)

In this Python code, a dictionary data = {'a': 10, 'b': 20, 'c': 30} is defined, and the variable sum is initialized to 0. The for loop iterates through the dictionary keys, adding each corresponding value to sum. In the first iteration, sum becomes 10, and in the second iteration, it becomes 30. In the third iteration, sum reaches 60, and since it exceeds 50, the break statement stops the loop. The final result printed is 60.
මෙම පයිතන් කේතයේ, dictionary යක් data = {'a': 10, 'b': 20, 'c': 30} අර්ථ දක්වා ඇති අතර, sum විචල්‍යය 0 න් ආරම්භ වේ. for ලූපයක් නැවත නැවත සිදු වේ. dictionary යතුරු හරහා, එක් එක් අනුරූප අගය sum වෙත එකතු කරයි. පළමු පුනරාවර්තනයේ දී, sum 10 බවට පත් වන අතර, දෙවන පුනරාවර්තනයේ දී, එය 30 බවට පත් වේ. තුන්වන පුනරාවර්තනයේ දී, sum 60 ට ළඟා වන අතර, එය 50 ඉක්මවන බැවින්, break ප්‍රකාශයෙන් ලූපය නතර කරයි. මුද්‍රණය කර ඇති අවසාන ප්‍රතිඵලය 60 වේ.

10. Which statement about the <em> and <strong> tags is correct?
<em> සහ <strong> උසුලන පිළිබඳ කුමන ප්‍රකාශය නිවැරදිද?

1) Both <em> and <strong> tags visually italicize text by default.
<em> සහ <strong> උසුලන දෙකම පෙරනිමියෙන් පාඨ දෘෂ්‍යව ඇල කරයි.

2) <u><em> is used for emphasizing text with visual effects, while <strong> indicates importance semantically.</u>
දෘශ්‍ය ප්‍රයෝග සහිත පෙළ අවධාරණය කිරීම සඳහා <em> භාවිත වන අතර, <strong> වැදගත්කම අර්ථකථනය කරයි.

3) Both <em> and <strong> only modify the visual appearance without semantic meaning.
<em> සහ <strong> යන දෙකම අර්ථකථන අර්ථයකින් තොරව දෘශ්‍ය පෙනුම පමණක් වෙනස් කරයි.

4) <em> and <strong> are replaced by <b> and <i> in modern HTML standards.
නවීන HTML ප්‍රමිතීන් තුළ <em> සහ <strong> <b> සහ <i> මගින් ප්‍රතිස්ථාපනය වේ.

5) <em> is preferred for underlined text, and <strong> for bold text.
යටින් ඉරි ඇඳ ඇති පෙළ සඳහා <em> වඩාත් හොඳ වන අතර තද පාඨ සඳහා <strong> වේ.

Answer : 2)

<em> (emphasis):
Adds emphasis to the text, typically displayed in italics by default. It conveys semantic importance by suggesting that the text should be read with focus.
පෙළට අවධාරණය එක් කරයි, සාමාන්‍යයෙන් පෙරනිමියෙන් ඇල අකුරු වලින් සංදර්ශන කෙරේ. පාඨය අවධානයෙන් යුතුව කියවිය යුතු බව යෝජනා කිරීමෙන් එය අර්ථකථන වැදගත්කමක් ලබා දෙයි.

<strong> (strong emphasis):
Indicates that the text is important or has strong emphasis, typically displayed in bold by default, but it also carries semantic meaning.
සාමාන්‍යයෙන් පෙරනිමියෙන් තද අකුරින් සංදර්ශණය වන, පෙළ වැදගත් හෝ දැඩි අවධාරණයක් ඇති බව අඟවයි, නමුත් එය අර්ථකථන අර්ථය ද දරයි.

Explanation of each option / එක් එක් විකල්පය පැහැදිලි කිරීම:

Option / විකල්ප 1)
Incorrect. <em> usually renders text in italics, but <strong> renders it in bold. They have distinct visual effects.
වැරදියි. <em> සාමාන්‍යයෙන් ඇල අකුරු වලින් පෙළ විදැහුම් කරයි, නමුත් <strong> එය තද අකුරින් ලබා දෙයි. ඒවාට වෙනම දෘශ්‍ය ප්‍රයෝග ඇත.

Option / විකල්ප 2)
Correct. <em> emphasizes text, often displayed in italics, while <strong> conveys importance, often displayed in bold. Both have semantic significance, aiding screen readers.
නිවැරදියි. <em> පෙළ අවධාරණය කරයි, බොහෝ විට ඇල අකුරු වලින් සංදර්ශණය වන අතර, <strong> වැදගත්කම ප්‍රකාශ කරයි, බොහෝ විට තද අකුරින් පෙන්වයි. දෙකටම අර්ථකථන වැදගත්කමක් ඇත, තිර කියවනයට උපකාරී වේ.

Option / විකල්ප 3)
Incorrect. These tags provide semantic meaning, helping assistive technologies understand the importance or emphasis of the text.
වැරදියි. මෙම උසුලන අර්ථකථන අර්ථයක් සපයන අතර, පාඨයේ වැදගත්කම හෝ අවධාරණය තේරුම් ගැනීමට උපකාරක තාක්ෂණයට උපකාර කරයි.

Option / විකල්ප 4)
Incorrect. <b> and <i> are purely for visual styling, while <em> and <strong> retain semantic meaning in modern HTML.
වැරදියි. <b> සහ <i> සම්පූර්ණයෙන්ම දෘශ්‍ය මෝස්තර සඳහා වන අතර, <em> සහ <strong> නවීන (HTML) හි අර්ථකථන අර්ථය රඳවා ගනී.

Option / විකල්ප 5)
Incorrect. `<em>` emphasizes text (usually italics), not underlined text. `<strong>` highlights importance, typically in bold.
වැරදියි. `<em>` අවධාරණය කරන්නේ පෙළ (සාමාන්‍යයෙන් ඇල අකුරු), යටින් ඉරි ඇඳ ඇති පෙළ නොවේ. `<strong>` වැදගත්කම උද්දීපනය කරයි, සාමාන්‍යයෙන් තද අකුරින්.

Therefore, the correct answer is 2).
එබැවින්, නිවැරදි පිළිතුර 2) වේ.