Information and Communication Technology

තොරතුරු හා සන්නිවේදන තාක්ෂණය

2026 QUIZ NO 487 MARKING

2026/04/11
ictfromabc
Ravindu Bandaranayake

1. Which of the following developments most directly enabled the transition from additive number systems to positional systems, marking a pivotal moment in the history of numerical abstraction?
සංඛ්‍යාත්මක වියුක්තකරණයේ ඉතිහාසයේ වැදගත් අවස්ථාවක් සනිටුහන් කරමින්, ආකලන සංඛ්‍යා පද්ධතිවල සිට ස්ථානීය පද්ධති දක්වා සංක්‍රමණය වීමට වඩාත් සෘජුවම ඉඩ සැලසූ පහත සඳහන් වර්ධනයන්ගෙන් කවරක් ද?

   1) The invention of Roman numerals by the ancient Romans.
   පුරාණ රෝමවරුන් විසින් රෝම ඉලක්කම් සොයා ගැනීම.
   2) The introduction of tally marks in prehistoric record-keeping.
   ප්‍රාග් ඓතිහාසික වාර්තා තබා ගැනීමේදී ගණන් කිරීමේ ලකුණු හඳුන්වාදීම.
   3) The creation of the base-60 system by the Babylonians.
   බැබිලෝනියානුවන් විසින් පාදම-60 පද්ධතිය නිර්මාණය කිරීම.
   4) The use of abaci in ancient Chinese commerce / පුරාණ චීන වාණිජ්‍යයේ අබාසි භාවිතය
   5) The adoption of the Hindu-Arabic numeral system in medieval Europe
   මධ්‍යකාලීන යුරෝපයේ හින්දු-අරාබි සංඛ්‍යා පද්ධතිය අනුගමනය කිරීම

Answer : 3)

Explanation / පැහැදිලි කිරීම:

Option / විකල්පය 1:
Incorrect. Roman numerals are a classic additive system. They lacked a place value concept, making arithmetic operations cumbersome and limiting abstraction.
වැරදියි. රෝම ඉලක්කම් යනු සම්භාව්‍ය ආකලන පද්ධතියකි. ඒවාට ස්ථාන අගය සංකල්පයක් නොතිබූ අතර, අංක ගණිතමය මෙහෙයුම් අපහසු සහ සීමාකාරී වියුක්තකරණයක් බවට පත් කළේය.

Option / විකල්පය 2:
Incorrect. Tally marks are a primitive additive representation with no concept of base or position. They reflect quantity, not mathematical structure.
වැරදියි. ගණන් කිරීමේ ලකුණු යනු පාදයක් හෝ පිහිටීමක් පිළිබඳ සංකල්පයක් නොමැති ප්‍රාථමික ආකලන නිරූපණයකි. ඒවා ගණිතමය ව්‍යුහය නොව ප්‍රමාණය පිළිබිඹු කරයි.

Option / විකල්පය 3:
Correct. The Babylonian base-60 (sexagesimal) system introduced a positional structure, where the value of a symbol depended on its place. Though it lacked a true zero, it paved the way for abstract numerical operations far beyond additive systems.
නිවැරදියි. බැබිලෝනියානු පාදම-60 (sexagesimal) පද්ධතිය ස්ථානීය ව්‍යුහයක් හඳුන්වා දුන් අතර, එහිදී සංකේතයක අගය එහි ස්ථානය මත රඳා පවතී. එයට සත්‍ය ශූන්‍යයක් නොතිබුණද, එය ආකලන පද්ධතිවලට වඩා බොහෝ දුරට වියුක්ත සංඛ්‍යාත්මක මෙහෙයුම් සඳහා මග පෑදීය.

Option / විකල්පය 4:
Incorrect. The abacus supported calculation, but it is a tool, not a written number system. It did not change the symbolic structure of numeric representation.
වැරදියි. ඇබකසය ගණනය කිරීමට සහාය වූ නමුත් එය මෙවලමක් මිස ලිඛිත සංඛ්‍යා පද්ධතියක් නොවේ. එය සංඛ්‍යාත්මක නිරූපණයේ සංකේතාත්මක ව්‍යුහය වෙනස් කළේ නැත.

Option / විකල්පය 5:
Incorrect. The Hindu-Arabic system was the culmination of positional logic, but not its origin. It refined and propagated positional ideas, especially through the concept of zero.
වැරදියි. හින්දු-අරාබි ක්‍රමය ස්ථානීය තර්කනයේ කුටප්‍රාප්තිය ව., නමුත් එහි ආරම්භය නොවේ. එය ස්ථානීය අදහස් පිරිපහදු කර ප්‍රචාරය කළේය, විශේෂයෙන් ශූන්‍ය සංකල්පය හරහා.

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

2. $89_{10} + 17_{10} = \ldots\ldots$

What is the binary value equivalent to the answer?
මෙහි පිළිතුරට තුල්‍ය වන ද්විමය අගය වන්නේ කුමක් ද?

1) 01101011 2) 01101010 3) 01100011 4) 01100010 5) 01111010

Answer: 2)

Explanation / පැහැදිලි කිරීම:

Here, 8910 + 1710 = 10610.
මෙහි, 8910 + 1710 = 10610 වේ.

Let's convert addition of 89 and 17 which is 106 into binary.
106 වන 89 සහ 17 හි එකතුව ද්විමය බවට පරිවර්තනය කරමු.

Converting 10610 into Binary 106_{10} ද්විමය බවට පරිවර්තනය කිරීම							
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1
0	1	1	0	1	0	1	0
01101010_2							

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

3. B. C + (A. (1. B + A. C) + C. C). D

Solve this Boolean expression.
මෙම බූලියානු ප්‍රකාශනය සුළු කරන්න.

1) $A.B + B.C + A.\overline{B}.$ 2) $\underline{B.C + A.B.D + D.C}$ 3) $A.\overline{B.D} + C.D$
4) $A.C.D + \overline{B}.$ 5) $\overline{C}.B + A.B.C + A.B$

Answer: 2)

This Boolean expression can be simplified as follows.
මෙම බූලියානු ප්‍රකාශනය පහත ආකාරයට සුළු කළ හැකිය.

$= BC + (A(1B + AC) + CC)D$	
$= BC + (A(B + AC) + CC)D$	Identity law / සර්වසාම්‍ය න්‍යාය
$= BC + (A(B + AC) + C)D$	Idempotent law / තද්භවාවී න්‍යාය
$= BC + DA(B + AC) + CD$	Distributive law / විෂටන න්‍යාය
$= BC + DAB + DAAC + CD$	Distributive law / විෂටන න්‍යාය
$= BC + DAB + DAC + CD$	Idempotent law / තද්භවාවී න්‍යාය
$= BC + DAB + CD + CDA$	Commutative law / න්‍යාදේශ න්‍යාය
$= BC + DAB + CD$	Redundancy law / සමතිරික්ත න්‍යාය

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

4. Which of the following transitions most likely involves long-term scheduling in an operating system?
පහත දැක්වෙන සංක්‍රාන්ති අතරින් බොහෝ විට මෙහෙයුම් පද්ධතියක දිගු කාලීන නියමකරණයට ඇතුළත් වන්නේ කුමක් ද?

   1) A process finishes its execution and terminates.
   ක්‍රියාවලියක් එහි ක්‍රියාත්මක කිරීම අවසන් කර අවසන් වේ.
   2) A process enters a waiting state for an I/O operation.
   ක්‍රියාවලියක් I/O මෙහෙයුමක් සඳහා පොරොත්තු තත්ත්වයකට ඇතුල් වේ.
   3) A new user logs in and starts a new application.
   නව පරිශීලකයෙකු ලොග් වී නව යෙදුමක් ආරම්භ කරයි.
   4) A process is interrupted by a higher-priority process.
   ඉහළ ප්‍රමුඛතා ක්‍රියාවලියක් මඟින් ක්‍රියාවලියකට බාධා ඇති වේ.
   5) A process is swapped out to secondary storage due to lack of memory.
   මතකය නොමැතිකම හේතුවෙන් ක්‍රියාවලියක් ද්විතියික ආචයනය වෙත මාරු කරනු ලැබේ.

Answer : 3)

In an operating system, scheduling is divided into three categories: long-term, short-term, and mid-term scheduling.
මෙහෙයුම් පද්ධතියක් තුළ, නියමකරණය කාණ්ඩ තුනකට බෙදා ඇත. දිගු කාලීන, කෙටි කාලීන සහ මධ්‍ය කාලීන නියමකරණය කිරීම.

Short-Term Scheduling / කෙටි කාලීන නියමකරණය (CPU Scheduling / CPU නියමකරණය)
Short-term scheduling, also known as CPU scheduling, is responsible for selecting which process from the ready queue will be assigned to the CPU next. This decision is made frequently, sometimes every few milliseconds, as the operating system works to maximize CPU utilization. Whenever the CPU becomes idle, the short-term scheduler selects the next process to execute.
කෙටි කාලීන නියමකරණය, CPU නියමකරණය ලෙසද හැඳින්වේ, සූදානම් පෝලිමේ සිට ඊළඟට CPU වෙත පැවරෙන්නේ කුමන ක්‍රියාවලියද යන්න තේරීම සඳහා වගකීම දරයි. CPU භාවිතය උපරිම කිරීමට මෙහෙයුම් පද්ධතිය ක්‍රියා කරන බැවින්, මෙම තීරණය නිතර, සමහර විට මිලි තත්පර කිහිපයකට වරක් ගනු ලැබේ. CPU ක්‍රියා විරහිත වන සෑම විටම, කෙටි කාලීන නියමකරණය ක්‍රියාත්මක කිරීමට ඊළඟ ක්‍රියාවලිය තෝරා ගනී.

Mid-Term Scheduling / මධ්‍ය කාලීන නියමකරණය (Swapping / මාරු කිරීම)
Mid-term scheduling focuses on managing system memory and controlling the level of multiprogramming, which refers to the number of processes that are currently loaded into memory. When system resources, particularly memory, become constrained, the mid-term scheduler will temporarily remove certain processes from memory and store them in secondary storage (virtual memory). This process is known as swapping.
මධ්‍ය කාලීන නියමකරණය පද්ධති මතකය කළමනාකරණය කිරීම සහ බහු ක්‍රමලේඛන මට්ටම පාලනය කිරීම කෙරෙහි අවධානය යොමු කරයි, එය දැනට මතකයට පටවා ඇති ක්‍රියාවලි ගණනට යොමු කරයි. පද්ධති සම්පත්, විශේෂයෙන්ම මතකය, සීමා වූ විට, මධ්‍ය කාලීන උපලේඛකයා විසින් යම් යම් ක්‍රියාවලි මතකයෙන් තාවකාලිකව ඉවත් කර ද්විතියික ගබඩාවේ (අථත්‍ය මතකය) ගබඩා කරනු ඇත. මෙම ක්‍රියාවලිය හුවමාරු කිරීම ලෙස හැඳින්වේ.

Long-Term Scheduling / දිගු කාලීන නියමකරණය (Job Scheduling / රැකියා නියමකරණය)
Long-term scheduling is responsible for controlling the admission of new processes into the system. It decides which processes should be moved from the job pool (secondary storage) into main memory for execution. This type of scheduling occurs less frequently than short-term or mid-term scheduling, as its main role is to regulate the system's workload by determining how many processes are in the ready queue.
පද්ධතියට නව ක්‍රියාවලි ඇතුළත් කිරීම පාලනය කිරීම සඳහා දිගුකාලීන නියමකරණය වගකිව යුතුය. එය ක්‍රියාත්මක කිරීම සඳහා රැකියා සංචිතයෙන් (ද්විතියික ගබඩාව) ප්‍රධාන මතකයට ගෙන යා යුතු ක්‍රියාවලින් තීරණය කරයි. මෙම ආකාරයේ නියමකරණ කෙටි කාලීන හෝ මධ්‍ය කාලීන නියමකරණවලට වඩා අඩුවෙන් සිදු වේ, එහි ප්‍රධාන කාර්යභාරය වන්නේ සූදානම් පෝලිමේ කොපමණ ක්‍රියාවලි තිබේද යන්න තීරණය කිරීමෙන් පද්ධතියේ කාර්ය භාරය නියාමනය කිරීමයි.

Therefore, among the given options, the most likely transition involving long-term scheduling is 3) A new user logs in and starts a new application.
එබැවින්, ලබා දී ඇති විකල්පයන් අතර, දිගු කාලීන නියමකරණය සම්බන්ධ බොහෝ දුරට ඉඩ ඇති සංක්‍රාන්තිය වේ 3) නව පරිශීලකයෙකු ලොග් වී නව යෙදුමක් ආරම්භ කරයි.

5. Given tables Employees and Departments with a foreign key dept_id in Employees referencing Departments, which SQL query correctly retrieves all employees and their respective department names, including employees without a department?
ලබා දී ඇති වගු, සේවකයින් යොමු කරන දෙපාර්තමේන්තු වල ආගන්තුක යතුර dept_id සහිත සේවකයින් සහ දෙපාර්තමේන්තු, විමසුමෙන් දෙපාර්තමේන්තුවක් නොමැති සේවකයින් ඇතුළුව සියලුම සේවකයින් සහ ඔවුන්ගේ අදාළ දෙපාර්තමේන්තු නම් නිවැරදිව ලබා ගැනීමට අදාළ නිවැරදි SQL කේතය වන්නේ කුමක්ද?

1) `SELECT Employees.name, Departments.dept_name FROM Employees INNER JOIN Departments ON Employees.dept_id = Departments.dept_id;`
2) `SELECT Employees.name, Departments.dept_name FROM Employees RIGHT JOIN Departments ON Employees.dept_id = Departments.dept_id;`
3) `SELECT Employees.name, Departments.dept_name FROM Employees LEFT JOIN Departments ON Employees.dept_id = Departments.dept_id;`
4) `SELECT Employees.name, Departments.dept_name FROM Employees FULL JOIN Departments ON Employees.dept_id = Departments.dept_id;`
5) `SELECT Employees.name, Departments.dept_name FROM Employees CROSS JOIN Departments ON Employees.dept_id = Departments.dept_id;`

Answer : 3)

This query uses a `left join`, which retrieves all employees, regardless of whether they have a matching department in the **`Departments`** table. For employees without a department, the **`dept_name`** will be **`NULL`**, ensuring that no employees are excluded from the results. This is the correct query to include all employees and their department names, including those without a department.
මෙම විමසුම `left join` භාවිතා කරයි, එය දෙපාර්තමේන්තු වගුවේ ගැලපෙන දෙපාර්තමේන්තුවක් තිබේද යන්න නොසලකා සියලුම සේවකයින් ලබා ගනියි. දෙපාර්තමේන්තුවක් නොමැති සේවකයින් සඳහා, දෙපාර්තමේන්තුවේ නම **`(dept_name) NULL`** වනු ඇත, එමඟින් කිසිදු සේවකයෙකු ප්‍රතිඵල වලින් බැහැර නොවන බව සහතික කෙරේ. දෙපාර්තමේන්තුවක් නොමැති අය ඇතුළුව සියලුම සේවකයින් සහ ඔවුන්ගේ දෙපාර්තමේන්තු නම් පෙන්වීමට මෙය නිවැරදි විමසුමයි.

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

6. Which of the following statements will correctly create a dictionary?
පහත සඳහන් කුමන ප්‍රකාශය නිවැරදිව dictionary එකක් නිර්මාණය කරයි ද?

1) `dict = {1, 2, 3}`
2) `dict = {"one": 1, "two": 2}`
3) `dict = [1: "one", 2: "two"]`
4) `dict = (1: "one", 2: "two")`
5) `dict = {1: "one", 2, "two"}`

Answer: 2)

Let's go through the given options / ලබා දී ඇති විකල්ප හරහා යමු.

1) `dict = {1, 2, 3}`:
This creates a set, not a dictionary. Sets are unordered collections of unique elements. There are no key-value pairs here, so it is not a valid dictionary.
මෙය dictionary එකක් නොව Set එකක් නිර්මාණය කරයි. Set යනු අනන්‍ය මූලාංගවල ඇණවුම් නොකළ එකතුවකි. මෙහි යතුරු අගය යුගල නොමැත. එබැවින්, එය වලංගු dictionary එකක් නොවේ.

2) `dict = {"one": 1, "two": 2}`:
This is the correct way to create a dictionary. The curly braces {} are used correctly, and it has key-value pairs where "one" and "two" are the keys, and 1 and 2 are the values.
Dictionary එකක් නිර්මාණය කිරීමේ නිවැරදි ක්‍රමය මෙයයි. සගල වරහන් {} නිවැරදිව භාවිතා කර ඇති අතර, එහි "one" සහ "two" යතුරු වන අතර 1 සහ 2 අගයන් වන යතුරු අගය යුගල ඇත.

3) `dict = [1: "one", 2: "two"]`:
This is incorrect. Square brackets [] are used for lists, not dictionaries. Dictionaries require curly braces {} for key-value pairs.
මෙය වැරදියි. කොටු වරහන් [] භාවිතා කරන්නේ ලැයිස්තු සඳහා මිස dictionaries සඳහා නොවේ. dictionary වලට යතුරු-අගය යුගල සඳහා සගල වරහන් {} අවශ්‍ය වේ.

4) `dict = (1: "one", 2: "two"):`
This is incorrect. Parentheses `()` are used for tuples, not dictionaries. A dictionary should be enclosed in curly braces `{}`.
මෙය වැරදියි. වරහන් :- භාවිතා කරන්නේ tuple සඳහා මිස dictionaries සඳහා නොවේ. dictionary එකක් සඟල වරහන් `{}` තුළ අමුණා තිබිය යුතුය.

5) `dict = {1: "one", 2, "two"}:`
This is incorrect due to a syntax error. The key-value pair 1: "one" is correct, but there is a mistake with 2, "two", which is not in a key-value format. There should be a colon separating the key and the value.
අක්ෂර දෝෂයක් නිසා මෙය වැරදියි. යතුරු අගය යුගලය 1: "one" නිවැරදියි, නමුත් 2, "two", සමඟ දෝෂයක් ඇත, එය ප්‍රධාන අගය ආකෘතියේ නැත. යතුර සහ අගය වෙන් කරන කෝලනයක් තිබිය යුතුය.

So, option 2) is the only valid way to create a dictionary.
එබැවින්, dictionary එකක් නිර්මාණය කිරීමට ඇති එකම වලංගු මාර්ගය විකල්ප 2) වේ.

7. Consider the following Python program.
පහත පයිතන් වැඩසටහන සලකා බලන්න.

```
player_file = open("playerData.txt", "r")
summary_file = open("summaryData.txt", "w")
threshold = 10
for line in player_file:
    if int(line.split()[1]) > threshold:
        summary_file.write(line)
player_file.close()
summary_file.close()
```

Which of the following statements are correct about the above program?
ඉහත වැඩසටහන සම්බන්ධයෙන් පහත සඳහන් කුමන ප්‍රකාශයන් නිවැරදි ද?

A- The program reads player data from playerData.txt one line at a time.
වැඩසටහන playerData.txt වෙතින් ක්‍රීඩක දත්ත වරකට එක පේළිය බැගින් කියවයි.

B- Player scores greater than 10 are written to summaryData.txt.
10ට වඩා වැඩි ක්‍රීඩකයන්ගේ ලකුණු summaryData.txt වෙත ලියා ඇත.

C- If playerData.txt does not exist, the program will raise an error at runtime.
playerData.txt නොපවතියි නම්, වැඩසටහන ධාවන වේලාවේදී දෝෂයක් මතු කරයි.

1) A only A පමණි
2) C only C පමණි
3) A and B only A හා B පමණි
4) A and C only A හා C පමණි
5) <u>All of the above</u> ඉහත සියල්ලම

Answer: 5)

Let's consider the code line by line / කේතය පේළියෙන් පේළිය සලකා බලමු:

`player_file = open("playerData.txt", "r"):`
Opens playerData.txt in read mode.
playerData.txt කියවීමේ ආකාරයෙන් විවෘත කරයි.

`summary_file = open("summaryData.txt", "w"):`
Opens summaryData.txt in write mode.
ලිවීමේ ආකාරයෙන් විවෘත කරයි.

`threshold = 10:`
Sets a threshold value of 10.
threshold අගය 10 සකසයි.

`for line in player_file:`
Reads the file playerData.txt line by line.
playerData.txt ගොනුව පේළියෙන් පේළිය කියවයි.

`if int(line.split()[1]) > threshold:`

Splits each line by whitespace and checks if the second element (index 1) converted to an integer is greater than 10.
සෑම පේළියක්ම අයසඤ්ඤ මගින් බෙදන අතර දෙවන මූලද්‍රව්‍යය (දර්ශක 1) පූර්ණ සංඛ්‍යාවකට පරිවර්තනය කර 10 ට වඩා වැඩි දැයි පරීක්ෂා කරයි.

`summary_file.write(line):`

Writes the line to summaryData.txt if the condition is met.
කොන්දේසිය සපුරා ඇත්නම් රේඛාව summaryData.txt වෙත ලියයි.

Closes both files using player_file.close() and summary_file.close().
player_file.close() සහ summary_file.close() භාවිතයෙන් ගොනු දෙකම වසා දමයි.

Analysis of statements / ප්‍රකාශ විශ්ලේෂණය:

Statement / ප්‍රකාශය A
True. The `for line in player_file:` statement reads the file line by line.
සත්‍ය වේ. `for line in player_file:` ප්‍රකාශය ගොනුව පේළියෙන් පේළිය කියවයි.

Statement / ප්‍රකාශය B
True. The condition `if int(line.split()[1]) > threshold:` checks if the second item in the line is greater than 10, and if true, the line is written to summaryData.txt.
සත්‍ය වේ. කොන්දේසිය `if int(line.split()[1]) > threshold:` පේළියේ දෙවන අයිතමය 10 ට වඩා වැඩි දැයි පරීක්ෂා කරයි, සහ සත්‍ය නම්, රේඛාව summaryData.txt වෙත ලියා ඇත.

Statement / ප්‍රකාශය C
True. If the file playerData.txt does not exist, trying to open it in read mode (`"r"`) will raise a `FileNotFoundError.`
සත්‍ය වේ. playerData.txt ගොනුව නොපවතියි නම්, එය read mode (`"r"`) විවෘත කිරීමට උත්සාහ කිරීම `FileNotFoundError` මතු කරයි.

The correct answer is 5) All of the above, as all the statements are true.
සියලුම ප්‍රකාශ සත්‍ය බැවින් නිවැරදි පිළිතුර 5) ඉහත සියල්ලම වේ.

8. Consider the below python code and answer the following questions.
පහත පයිතන් කේතය සලකා පහත ප්‍රශ්න වලට පිළිතුරු සපයන්න.

```
Cost = int (input ('Enter a Cost in LKR '))
X = (5000, 2000, 1000, 500, 100, 50, 20, 10, 5, 2, 1)
Y , Count = True , 0
while Cost > 0 :
  if Cost >= X[Count] :
     print(X[Count], end = '|')
     Cost = Cost - X[Count]
  else:
     Count = Count + 1
```

What is the primary function of the above Python code? / පහත පයිතන් කේතයේ මූලික අරමුණ කුමක් ද?

1) To calculate the total number of items a person can buy with the entered cost.
ඇතුළත් කළ පිරිවැය සමග පුද්ගලයෙකුට මිලදී ගත හැකි මුළු අයිතම ගණන ගණනය කිරීම.
2) To break down a given cost into the largest possible currency denominations and print them.
දී ඇති පිරිවැයක් හැකි විශාලතම මුදල් වර්ග වලට කඩා ඒවා මුද්‍රණය කිරීම.
3) To compare the cost entered by the user with predefined values and give suggestions.
පරිශීලක විසින් ඇතුළත් කරන පිරිවැය පූර්ව නිශ්චිත අගයන් සමග සංසන්දනය කර යෝජනා ලබා දීම.
4) To determine the exact number of coins needed to make a specific amount.
නිශ්චිත මුදලක් සෑදීමට අවශ්‍ය කාසි නිශ්චිත සංඛ්‍යාව තීරණය කිරීම.
5) To convert the cost into another currency using a fixed exchange rate.
ස්ථාවර විනිමය අනුපාතයක් භාවිතා කරමින් පිරිවැය වෙනත් මුදල් ඒකකයකට පරිවර්තනය කිරීම.

Answer : 4)

Explanation of the code / කේතය පැහැදිලි කිරීම.

`Cost = int(input('Enter a Cost in LKR '))`
This line prompts the user to enter a cost in Sri Lankan Rupees (LKR) and converts it into an integer.
මෙම රේඛාව ශ්‍රී ලංකා රුපියල්වලින් (LKR) පිරිවැයක් ඇතුළත් කිරීමට පරිශීලකයා පොළඹවන අතර එය පූර්ණ සංඛ්‍යාවක් බවට පරිවර්තනය කරයි.

`X = (5000, 2000, 1000, 500, 100, 50, 20, 10, 5, 2, 1)`
X is a tuple containing currency in descending order, from LKR 5000 down to LKR 1. The program will use these values to break down the cost.
X යනු LKR 5000 සිට LKR 1 දක්වා අවරෝහණ අනුපිළිවෙලෙහි මුදල් අගයන් අඩංගු tuple එකකි. වැඩසටහන පිරිවැය බිඳ දැමීමට මෙම අගයන් භාවිතා කරනු ඇත.

`Y, Count = True, 0`
Y is set to True (though it's unused and could be removed with no effect on the code).
Y සත්‍ය ලෙස සකසා ඇත (එය භාවිතයට නොගත් අතර කේතයට කිසිදු බලපෑමක් නොමැතිව ඉවත් කළ හැක).

Count is set to 0 and will be used as an index to access the currency in X.
Count, 0 ලෙස සකසා ඇති අතර X හි මුදල් අගයන් වෙත ප්‍රවේශ වීමට දර්ශකයක් ලෙස භාවිතා කරනු ඇත.

`while Cost > 0:`
This loop will continue until Cost becomes zero.
පිරිවැය ශූන්‍ය වන තෙක් මෙම ලූපය දිගටම පවතිනු ඇත.

```
if Cost >= X[Count]:
  print(X[Count], end='|')
```

If the remaining Cost is greater than or equal to the current number of notes or coins (`X[Count]`), the program:
ඉතිරි පිරිවැය වත්මන් කාසි හෝ නෝට්ටු ගණන (`X[Count]`) වඩා වැඩි හෝ සමාන නම්, වැඩසටහන:

Prints the number of notes or coins / කාසි හෝ නෝට්ටු ගණන මුද්‍රණය කරයි
Subtracts this number of notes or coins from Cost / මෙම කාසි හෝ නෝට්ටු ගණන පිරිවැයෙන් අඩු කරයි

```
else:
Count = Count + 1
```

If Cost is smaller than `X[Count]`, it increments Count to check the next smaller denomination.
පිරිවැය `X[Count]` ට වඩා කුඩා නම්, එය මීළඟ කුඩා කාසි හෝ නෝට්ටු ගණන පිරික්සීමට Count එක වැඩි කරයි.

The code's primary function is to determine the exact number of coins or notes needed to make a specific amount.
කේතයේ මූලික කාර්යය වන්නේ නිශ්චිත මුදලක් සෑදීමට අවශ්‍ය කාසි හෝ නෝට්ටු නිශ්චිත සංඛ්‍යාව තීරණය කිරීම.

So, the correct answer is 4).
එබැවින්, නිවැරදි පිළිතුර 4) වේ.

9. A form requires the submit button to be highlighted when a user hovers over it or focuses on it via keyboard navigation.
පරිශීලකයෙකු එය මත සැරිසරන විට හෝ යතුරුපුවරු සංචාලනය භරහා ඒ වෙත අවධානය යොමුකරන විට පෝරමයකට ඉඳිරිපත් කිරීමේ බොත්තම උද්දීපනය කිරීම අවශ්‍ය වේ.

Which CSS selectors should be used / කුමන CSS තේරීම් භාවිතා කළ යුතු ද?

1) `button:visited { background-color: #4CAF50; }`
2) `button:active { background-color: #4CAF50; }`
3) `button:hover, button:focus { background-color: #4CAF50; }`
4) `input[type=submit]:checked { background-color: #4CAF50; }`
5) `button:target { background-color: #4CAF50; }`

Answer : 3)

Explanation / පැහැදිලි කිරීම:

Option / විකල්ප 2
`:active` applies only while the button is being clicked. It does not remain highlighted when hovering or when focused via keyboard.
`:active` බොත්තම ක්ලික් කරන අතරතුර පමණක් අදාළ වේ. එය සැරිසරන විට හෝ යතුරුපුවරුව භරහා අවධානය යොමු කරන විට උද්දීපනය නොවී පවතී.

Option / විකල්ප 1
`:visited` applies only to links (`<a>` elements) that the user has visited. It does not apply to buttons at all.
`:visited` අදාළ වන්නේ පරිශීලකයා පිවිසි සබැඳි (`<a>` elements) සඳහා පමණි. එය බොත්තම් සඳහා කිසිසේත්ම අදාළ නොවේ.

Option / විකල්ප 3
`:hover` applies when the mouse pointer is over the button, and `:focus` applies when the button is selected via keyboard (Tab key). Combining both ensures accessibility and interactivity, which perfectly matches the requirement.
මූසික දර්ශකය බොත්තම මත ඇති විට `:hover` අදාළ වන අතර, යතුරුපුවරුව (ටැබ් යතුර) භරහා බොත්තම තෝරාගත් විට `:focus` අදාළ වේ. දෙකම ඒකාබද්ධ කිරීම ප්‍රවේශ්‍යතාව සහ අන්තර්ක්‍රියාකාරීත්වය සහතික කරයි, එය අවශ්‍යතාවයට හොඳින් ගැලපේ.

Option / විකල්ප 4
`:checked` applies to checkboxes or radio buttons, not submit buttons. So it cannot highlight a standard submit button.
`:checked` අදාළ වන්නේ වෙක් කොටු හෝ රේඩියෝ බොත්තම් සඳහා මිස ඉදිරිපත් කිරීමේ බොත්තම් සඳහා නොවේ. එබැවින් එයට සම්මත ඉදිරිපත් කිරීමේ බොත්තමක් ඉස්මතු කළ නොහැක.

Option / විකල්ප 5
`:target` applies to elements targeted via URL fragment identifiers (Like example.com/page#section). It has no effect on button hover or focus.
`:target` කොටස් හඳුනාගැනීම් හරහා ඉලක්ක කරගත් මූලද්‍රව්‍ය සඳහා අදාළ වේ (example.com/page#section). . එය බොත්තම් හොවර් කිරීම හෝ නාභිගත කිරීම කෙරෙහි කිසිදු බලපෑමක් ඇති නොකරයි.

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

10. Which is the correct way to insert the submit button in form handling in the way depicted below?
පහත සඳහන් ආකාරයට ගොනු හැසිරවීමේදී submit බොත්තම යොදන්නේ කෙසේ ද?

1) `<input type="submit" name="submit">`
2) `<submit value="save">`
3) `<input type="submit" value="submit">`
4) `<input type="submit" name="submit" value="save">`
5) `<submit name="submit" value="save">`

Answer: 4)

Explanation / පැහැදිලි කිරීම:

The correct way to insert the submit button in form handling, as depicted in the image, is 4) `<input type="submit" name="submit" value="save">`
රූපයේ දැක්වෙන පරිදි, පෝරම හැසිරවීමේදී ඉදිරිපත් කිරීමේ බොත්තම ඇතුළත් කිරීමේ නිවැරදි ක්‍රමය 4) `<input type="submit" name="submit" value="save">` වේ.

Here's why this is correct / මෙය නිවැරදි වන්නේ මන්දැයි මෙන්න:

`type="submit":`
Specifies that the input element is a submit button.
ආදාන මූලාංගය ඉදිරිපත් කිරීමේ බොත්තමක් බව සඳහන් කරයි.

`name="submit":`
Assigns a name to the button, which can be used to identify the button in the server-side script when the form is submitted.
බොත්තමට නමක් පවරයි, පෝරමය ඉදිරිපත් කරන විට සේවාදායක පැත්තේ ස්ක්‍රිප්ටයේ ඇති බොත්තම හඳුනා ගැනීමට එය භාවිතා කළ හැක.

`value="save":`
Sets the text displayed on the button. In this case, it displays "save" on the button, matching what is depicted in the image.
බොත්තම මත දිස්වන පෙළ සකසයි. මෙම අවස්ථාවේදී, එය රූපයේ දැක්වෙන දේට ගැළපෙන බොත්තම මත "save" පෙන්වයි.