සියලු ම හිමිකම් ඇවිරිණි / *All Rights Reserved*]

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

Information and Communication Technology

තොරතුරු හා සන්නිවේදන තාක්ෂණය

2026 QUIZ NO 219 MARKING

2025/07/17
ictfromabc
Ravindu Bandaranayake

1. The invention of the "3 pin vacuum tube", laid the foundation for the modern computer technology. The person who is honored for this is ?
"අග්‍ර තුනේ රික්තක නළය" සොයාගැනීම, නවීන පරිගණක තාක්ෂණය සඳහා අඩිතාලම දැමීය. මේ සඳහා ගෞරවයට පාත්‍ර වූ පුද්ගලයා කවුද?

   1) John Presper Eckert / ජෝන් ප්‍රෙස්පර් එකර්ට්.
   2) Lee De Forest / ලී ද ෆොරස්ට්.
   3) John V Atanasoff / ජෝන් වී ඇටනසොෆ්.
   4) Gottfried Wilhelm / ගොට්ෆ්‍රිඩ් විල්හෙල්ම්.
   5) Howard Aiken / හෝවාර්ඩ් අයිකන්.

   Answer: 2)

   Explanation / පැහැදිලි කිරීම:

   Lee De Forest is credited with the invention of the vacuum tube (also known as the electronic valve). His work on the vacuum tube was crucial in the development of early electronic computers, as it allowed for the amplification and switching of electronic signals, which was essential for the functioning of these machines.
   Lee De Forest රික්තක නළය (ඉලෙක්ට්‍රොනික කපාටය ලෙසද හැඳින්වේ) සොයාගැනීමේ ගෞරවය හිමි වේ. මෙම යන්ත්‍රවල ක්‍රියාකාරිත්වය සඳහා අත්‍යාවශ්‍ය වූ විද්‍යුත් සංඥා විස්තාරණය කිරීමට සහ මාරු කිරීමට එය ඉඩ සලසා දුන් බැවින්, රික්ත නළය පිළිබඳ ඔහුගේ කාර්යය මුල් කාලීන ඉලෙක්ට්‍රොනික පරිගණක සංවර්ධනය කිරීමේදී තීරණාත්මක විය.

   Why the other options are incorrect / අනෙක් විකල්ප වැරදි ඇයි.

   1) John Presper Eckert
   Along with John Mauchly, Eckert is known for designing the ENIAC, one of the earliest electronic general-purpose computers, but he did not invent the electronic valve.
   John Mauchly සමඟ එක්ව Eckert පැරණිතම ඉලෙක්ට්‍රොනික පොදු කාර්ය පරිගණක වලින් එකක් වන ENIAC නිර්මාණය කිරීම සම්බන්ධයෙන් ප්‍රසිද්ධියක් උසුලන නමුත් ඔහු ඉලෙක්ට්‍රොනික කපාටය සොයා ගත්තේ නැත.

   3) John V. Atanasoff
   Known for creating the Atanasoff-Berry Computer (ABC), which was an early electronic computer, but he did not invent the electronic valve.
   මුල් කාලයේ ඉලෙක්ට්‍රොනික පරිගණකයක් වූ Atanasoff-Berry Computer (ABC) නිර්මාණය කිරීම සඳහා ප්‍රසිද්ධියට පත් වූ නමුත් ඔහු ඉලෙක්ට්‍රොනික කපාටය සොයා ගත්තේ නැත.

   4) Gottfried Wilhelm Von Leibniz
   A philosopher and mathematician known for his work in calculus and binary number systems, not for the invention of the electronic valve.
   දාර්ශනිකයෙක් සහ ගණිතඥයෙක් ඉලෙක්ට්‍රොනික කපාටය සොයා ගැනීම සඳහා නොව කලනය සහ ද්විමය සංඛ්‍යා පද්ධති පිළිබඳ ඔහුගේ වැඩ සඳහා ප්‍රසිද්ධය.

   5) Howard Aiken
   An American physicist and computer pioneer known for the Harvard Mark I computer, but he did not invent the electronic valve.
   Harvard Mark I පරිගණකය සඳහා ප්‍රසිද්ධ ඇමරිකානු භෞතික විද්‍යාඥයෙක් සහ පරිගණක පුරෝගාමියෙක්, නමුත් ඔහු ඉලෙක්ට්‍රොනික කපාටය සොයා ගත්තේ නැත.

   Therefore, the correct answer is 2) Lee De Forest
   එබැවින්, නිවැරදි පිළිතුර 2) Lee De Forest ය.

2. 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.

1) Magnetic Drum / චුම්භක හර
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.

2) 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.

3) 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.

4) 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.

5) 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.

3. Among these, the number of bits possible for Sign, Exponent, and Mantissa in IEEE 754 - Double Precision is,
මේ අතරින් IEEE 754 - Double Precision හි Sign, Exponent සහ Mantissa සඳහා තිබිය හැකි බිටු ගණන වන්නේ,

1) 1, 11, 32 2) 1, 7, 24 3) 1, 8, 23 4) 1, 11, 52 5) 1, 12, 33

Answer: 4)

Explanation / පැහැදිලි කිරීම:

IEEE 754 Double Precision Format / IEEE 754 ද්විත්ව නිරවද්‍යතා ආකෘතිය

IEEE 754 double precision format, also known as binary64, represents floating-point numbers using three components: a sign bit, an exponent, and a mantissa (also known as the significand). The total number of bits used is 64

IEEE 754 ද්විත්ව නිරවද්‍යතා ආකෘතිය, binary64 ලෙසද හැඳින්වේ, සංරචක තුනක් භාවිතා කරමින් floating-point සංඛ්‍යා නියෝජනය කරයි: sign bit, exponent සහ a mantissa (significand) ලෙසද හැඳින්වේ. භාවිතා කරන ලද මුළු බිටු ගණන 64 කි.

1) 1, 11, 32

In this option, the sign bit is correctly assigned 1 bit. However, the exponent and mantissa bits do not add up correctly to 64 bits. The exponent is given 11 bits (which is correct), but the mantissa is given only 32 bits, resulting in a total of 44 bits when combined with the sign bit, which is incorrect.

මෙම විකල්පය තුළ, සලකුණු බිටු නිවැරදිව 1 බිටු ලබා දී ඇත. කෙසේ වෙතත්, exponent සහ mantissa බිටු බිටු 64 දක්වා නිවැරදිව එකතු නොවේ. ඝාතකයට බිටු 11 ක් ලබා දී ඇත (එය නිවැරදියි), නමුත් mantissa ලබා දී ඇත්තේ බිටු 32ක් පමණි, එහි ප්‍රතිඵලය ලෙස ලකුණු බිටු සමඟ එකතු වූ විට සම්පූර්ණ බිටු 44ක් ලැබේ, එය වැරදිය.

2) 1, 7, 24

In this option, the sign bit is correctly assigned 1 bit. However, the exponent is given 7 bits, and the mantissa 24 bits, totaling only 32 bits, which is the format for single precision, not double precision.

මෙම විකල්පය තුළ, සලකුණු බිටු නිවැරදිව 1 බිටු ලබා දී ඇත. කෙසේ වෙතත්, exponent ට බිටු 7ක් ලබා දී ඇති අතර, mantissa ට බිටු 24ක් ලබා දී ඇති අතර, එය බිටු 32ක් පමණි, එය ද්විත්ව නිරවද්‍යතාවයක් නොව තනි නිරවද්‍යතාව සඳහා වන ආකෘතියයි.

3) 1, 8, 23

This option correctly assigns 1 bit for the sign. However, 8 bits for the exponent and 23 bits for the mantissa represent the IEEE 754 single precision format, not double precision, totaling 32 bits.

මෙම විකල්පය ලකුණ සඳහා බිටු 1ක් නිවැරදිව පවරයි. කෙසේ වෙතත්, exponent සඳහා බිටු 8ක් සහ mantissa සඳහා බිටු 23ක් නියෝජනය කරන්නේ IEEE 754 තනි නිරවද්‍යතා ආකෘතිය මිස ද්විත්ව නිරවද්‍යතාවයක් නොව, මුළු බිටු 32කි.

4) 1, 11, 52

This option correctly assigns 1 bit for the sign, 11 bits for the exponent, and 52 bits for the mantissa. This adds up to 64 bits in total, which is the correct format for IEEE 754 double precision representation. The 11 bits for the exponent allow for a range of values, and the 52 bits for the mantissa provide the precision needed for double precision numbers.

මෙම විකල්පය නිවැරදිව ලකුණ සඳහා බිටු 1 ක්, exponent සඳහා බිටු 11 ක් සහ mantissa සඳහා බිටු 52 ක් නියම කරයි. මෙය IEEE 754 ද්විත්ව නිරවද්‍යතා නිරූපණය සඳහා නිවැරදි ආකෘතිය වන මුළු බිටු 64ක් දක්වා එකතු කරයි. exponent සඳහා වන බිටු 11 අගයන් පරාසයකට ඉඩ සලසයි, සහ mantissa සඳහා බිටු 52 ද්විත්ව නිරවද්‍යතා සංඛ්‍යා සඳහා අවශ්‍ය නිරවද්‍යතාවය සපයයි.

5) 1, 12, 33

In this option, the sign bit is correctly assigned 1 bit. However, 12 bits for the exponent and 33 bits for the mantissa add up to 46 bits, which is not the correct total for double precision (64 bits).

මෙම විකල්පය තුළ, සලකුණු බිටි නිවැරදිව 1 බිටි ලබා දී ඇත. කෙසේ වෙතත්, exponent සඳහා බිටු 12 ක් සහ mantissa සඳහා බිටු 33 ක් බිටු 46 ක් දක්වා එකතු වේ, එය ද්විත්ව නිරවද්‍යතාවය බිටු 64 සඳහා නිවැරදි එකතුව නොවේ.

So, the correct configuration for the IEEE 754 double precision format is 1 bit for the sign, 11 bits for the exponent, and 52 bits for the mantissa, which makes option 4 the correct answer.

එබැවින්, IEEE 754 ද්විත්ව නිරවද්‍යතා ආකෘතිය සඳහා නිවැරදි වින්‍යාසය ලකුණ සඳහා බිටි 1 ක්, exponent සඳහා බිටු 11 ක් සහ mantissa සඳහා බිටු 52 ක් වන අතර එමඟින් විකල්පය 4 නිවැරදි පිළිතුර වේ.

4. Which of the following is the correct output for Bitwise XOR between two binary numbers 11010001_2 and 01101111_2?
11010001_2 හා 01101111_2 අතර බිටුඅනුසාරිත XOR සඳහා නිවැරදි ප්‍රතිදානය වන්නේ පහත සඳහන් කවරක්ද?

1) $\underline{10111110_2}$ 2) 00001110_2 3) 10100101_2 4) 11111111_2 5) 01000001_2

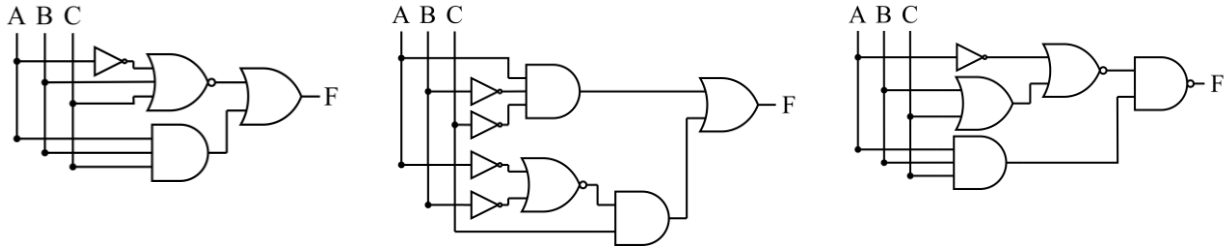
Answer: 1)

Explanation / පැහැදිලි කිරීම:

We compare each pair of bits to determine the correct output for the Bitwise XOR between the binary numbers 11010001_2 and 01101111_2. The XOR operation results in 1 if the bits are different and 0 if they are the same. By performing this operation on each pair of bits from the two binary numbers, we find that the resulting binary number is 10111110_2.
ද්විමය අංක 11010001_2 සහ 01101111_2 අතර බිටුඅනුසාරිතXOR සඳහා නිවැරදි ප්‍රතිදානය තීරණය කිරීම සඳහා අපි සෑම බිටු යුගලයක්ම සංසන්දනය කරමු. XOR මෙහෙයුමේ ප්‍රතිඵලය වන්නේ බිටු වෙනස් නම් 1 සහ ඒවා සමාන නම් 0 වේ. ද්විමය සංඛ්‍යා දෙකෙන් සෑම බිටු යුගලයක් මතම මෙම ක්‍රියාකාරකම සිදුකිරීමෙන්, ලැබෙන ද්විමය සංඛ්‍යාව 10111110_2 බව අපට පෙනී යයි.

Therefore, the correct output is option 1).
එබැවින් නිවැරදි ප්‍රතිදානය විකල්පය 1 වේ.

5. Which of the following circuits does the same logic apply?
පහත පරිපථ අතරින් එක සමාන තර්කයක් ක්‍රියාත්මක වන්නේ කුමන පරිපථ අතරද?



A B C

1) $\underline{\text{A and B only}}$.
A හා B පමණි.
2) A only.
A පමණි.
3) B only.
B පමණි.
4) B and C only.
B හා C පමණි.
5) All of the above.
ඉහත සියල්ලම.

Answer: 1)

Let's get the answer below. / පහත පරිදි පිළිතුර ලබාගනිමු.

A. $\overline{(\bar{A} + B + C)} + (ABC)$

$\overline{(\bar{A} + B + C)} + (ABC)$
$\bar{\bar{A}}\bar{B}\bar{C} + ABC$ De morgan's Law / ද මුවාර් න්‍යාය
$A\bar{B}\bar{C} + ABC$ Double Inverse Law / ද්විත්ව ප්‍රතිලෝම න්‍යාය

B. $(A\bar{B}\bar{C}) + (\overline{\bar{A} + \bar{B}})C$

$(A\bar{B}\bar{C}) + (\overline{\bar{A} + \bar{B}})C$
$A\bar{B}\bar{C} + \bar{\bar{A}}\bar{\bar{B}}C$ De morgan's Law / ද මුවාර් න්‍යාය
$A\bar{B}\bar{C} + A\bar{\bar{B}}C$ Double Inverse Law / ද්විත්ව ප්‍රතිලෝම න්‍යාය
$A\bar{B}\bar{C} + ABC$ Double Inverse Law / ද්විත්ව ප්‍රතිලෝම න්‍යාය

C. $\overline{\overline{(\overline{A}+B+C)}(ABC)}$

$\overline{\overline{(\overline{A}+B+C)}(ABC)}$

$\overline{\overline{\overline{A}+B+C}}+\overline{A}+\overline{B}+\overline{C}$ De morgan's Law / ද මුවාර් න්‍යාය

$\overline{A}+B+C+\overline{A}+\overline{B}+\overline{C}$ Double Inverse Law / ද්විත්ව ප්‍රතිලෝම න්‍යාය

$\overline{A}+C+\overline{A}+\overline{C}+1$ Inverse Law / ප්‍රතිලෝම න්‍යාය

1 Identity Law / සර්වසාම්‍ය න්‍යාය

So, the answer is 1).
එබැවින්, පිළිතුර 1) වේ.

6. What is the final answer obtained by simplifying the following Boolean expression?
පහත බූලිය ප්‍රකාශනය සුලු කිරීමෙන් ලැබෙන අවසාන පිළිතුර කුමක් ද?

$ABC+\overline{A}BC+AB\overline{C}$

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

Answer: 4)

Let's simplify the given Boolean expression.
ලබා දී ඇති බූලියන් ප්‍රකාශනය සරල කරමු.

$= ABC + \overline{A}BC + AB\overline{C}$

$= BC(A + \overline{A}) + AB\overline{C}$ Distributive Law / විෂටන න්‍යාය

$= BC(1) + AB\overline{C}$ Complement Law / ප්‍රතිලෝම න්‍යාය

$= BC + AB\overline{C}$ Identity Law / සර්වසාම්‍ය න්‍යාය

$= B(C + A\overline{C})$ Distributive Law / විෂටන න්‍යාය

$= B(A + C)$ Redundancy Law / සමතිරික්ත න්‍යාය

$= BA + BC$ Distributive Law / විෂටන න්‍යාය

Therefore, the correct output is option 4).
එබැවින් නිවැරදි ප්‍රතිදානය විකල්පය 4) වේ.

7. How many full and half adders are required to add a 16-bit number?
බිටු 16ක අංකයක් එකතු කිරීමට පූර්ණ සහ අර්ධ ආකලක කොපමණක් අවශ්‍ය වේද?

1) 8 half adders , 8 full adders / අර්ධ ආකලක 8 පූර්ණ ආකලක 8
2) <u>1 half adders , 15 full adders / අර්ධ ආකලක 1 පූර්ණ ආකලක 15</u>
3) 16 half adders , 0 full adders / අර්ධ ආකලක 16 පූර්ණ ආකලක 0
4) 4 half adders , 12 full adders / අර්ධ ආකලක 4 පූර්ණ ආකලක 12
5) 0 half adders , 16 full adders / අර්ධ ආකලක 0 පූර්ණ ආකලක 16

Answer: 2)

Explanation / පැහැදිලි කිරීම:

To add two 16-bit numbers, each digit must be added together along with any carry from the previous digit. A full adder can handle adding two bits along with a carry-in, producing a sum bit and a carry-out. However, for the least significant bit (LSB), a half adder is sufficient as there is no carry-in from a previous operation. For the remaining 15 bits, full adders are required to manage both the sum and any carry from the previous addition.
16-බිට් සංඛ්‍යා දෙකක් එකතු කිරීමට, එක් එක් ඉලක්කම් පෙර ඉලක්කමෙන් ඕනෑම රැගෙන යාමක් සමඟ එකට එකතු කළ යුතුය. පූර්ණ ආකලයකට රැගෙන යාමක් සමඟ බිටු දෙකක් එකතු කිරීම හැසිරවිය හැක, එකතුවක් සහ ගෙනයාමක් නිෂ්පාදනය කරයි. කෙසේ වෙතත්, අවම සැලකිය යුතු බිට් LSB සඳහා, පෙර මෙහෙයුමකින් ගෙනයාමක් නොමැති බැවින් අර්ධ එකතු කරන්නෙකු ප්‍රමාණවත් වේ. ඉතිරි බිටු 15 සඳහා, එකතුව සහ පෙර එකතු කළ ඕනෑම රැගෙන යාම යන දෙකම කළමනාකරණය කිරීමට සම්පූර්ණ එකතු කරන්නන් අවශ්‍ය වේ.

Therefore, to add 16-bit numbers, a combination of 1 half adder and 15 full adders is necessary to ensure accurate addition with consideration for carry propagation throughout the operation.
එබැවින්, 16 බිට් සංඛ්‍යා එකතු කිරීම සඳහා, මෙහෙයුම පුරාවට ගෙනයාමේ ප්‍රවාරණය සැලකිල්ලට ගනිමින් නිවැරදි එකතු කිරීම සහතික කිරීම සඳහා අර්ධ ආකලයකට 1 ක් සහ පූර්ණ ආකලයකට 15 ක එකතුවක් අවශ්‍ය වේ.

So, answer number 2 is correct.
ඒ අනුව, පිලිතුරු අංක 2 නිවැරදි වේ.

8. What is the output of the following Python code?
පහත පයිතන් කේතයේ ප්‍රතිදානය කුමක්ද?

```
print(type(42))
```

1) `<class 'int'>`
2) `<class 'str'>`
3) `<class 'float'>`
4) `<class 'bool'>`
5) `<class 'complex'>`

Answer: 1)

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

1) `<class 'int'>` (Correct)
The number `42` is an integer, and the `type()` function identifies it as an object of type `int`. Therefore, the correct output will be `<class 'int'>`.
අංක `42` යනු පූර්ණ සංඛ්‍යාවක් වන අතර `type()` ශ්‍රිතය එය `int` වර්ගයේ වස්තුවක් ලෙස හඳුනා ගනී. එබැවින් නිවැරදි ප්‍රතිදානය `<class 'int'>`.

2) `<class 'str'>`
The number `42` is not a string (`str`), so this type would not be returned. A string would be something like `"42"`, which would yield `<class 'str'>`.
අංක `42` string (`str`) එකක් නොවේ, එබැවින් මෙම වර්ගය ආපසු ලබා නොදෙනු ඇත. string එකක් `42` වැනි දෙයක් වනු ඇත, එය `<class 'str'>`.ලබා දෙනු ඇත.

3) `<class 'float'>`
`42` is an integer, not a floating-point number. A floating-point number would be written as `42.0`, and only then would the type returned be `<class 'float'>`.
`42` යනු පූර්ණ සංඛ්‍යාවක් මිස floating-point සංඛ්‍යාවක් නොවේ. floating-point අංකයක් `42.0` ලෙස ලියා ඇති අතර, ආපසු එන වර්ගය `<class 'float'>` වනු ඇත.

4) `<class 'bool'>`
A boolean value in Python is either `True` or `False`. `42` is'nt a boolean value, so this option is incorrect.
Python හි boolean `True` සත්‍ය හෝ `False` වේ. `42` යනු boolean අගයක් නොවේ, එබැවින් මෙම විකල්පය වැරදියි.

5) `<class 'complex'>`
A complex number in Python would look like `42 + 0j`. Since `42` is a simple integer and not a complex number, this type is incorrect.
Python හි සංකීර්ණ සංඛ්‍යාවක් ලෙස `42 + 0j` පෙනෙනු ඇත. `42` යනු සරල පූර්ණ සංඛ්‍යාවක් මිස සංකීර්ණ සංඛ්‍යාවක් නොවන බැවින්, මෙම වර්ගය වැරදිය.

So, the correct answer is 1).
එබැවින්, නිවැරදි පිළිතුර 1) වේ.

9. How do you create a variable with the numeric value 5?
සංඛ්‍යාත්මක අගය 5 වන විචල්‍යයක් සාදා ගන්නේ කෙසේද?

1) `num = int(5)` 2) `num = 5` 3) `num == 5` 4) `num : 5` 5) `int num = 5`

Answer: 2)

Let's look at each option. / එක් එක් විකල්පය දෙස බලමු.

1) `num = int(5).`
According to this option, it is false. While this syntax does correctly convert `5` to an integer, it is unnecessary to explicitly use `int()` to assign a numeric value. The simpler and more direct method is just assigning the value directly.
මෙම විකල්පය අනුව, එය අසත්‍ය වේ. මෙම වාක්‍ය ඛණ්ඩය නිවැරදිව `5` පූර්ණ සංඛ්‍යාවක් බවට පරිවර්තනය කරන අතර, සංඛ්‍යාත්මක අගයක් පැවරීම සඳහා පැහැදිලිවම `int()` භාවිතා කිරීම අනවශ්‍ය වේ. සරල හා වඩා සෘජු ක්‍රමය වන්නේ අගය කෙලින්ම පැවරීමයි.

2) `num = 5.`
According to this option, it is true. This is the correct way to create a variable with the numeric value `5` in Python. The syntax directly assigns the integer `5` to the variable `num`, which is the simplest and most common method.
මෙම විකල්පය අනුව, එය සත්‍ය වේ. Python හි සංඛ්‍යාත්මක අගය `5` සමඟ විචල්‍යයක් සෑදීමේ නිවැරදි ක්‍රමය මෙයයි. වාක්‍ය ඛණ්ඩය සෘජුවම `num` විචල්‍යයට පූර්ණ සංඛ්‍යාව `5` ලබා දෙයි, එය සරලම සහ වඩාත් පොදු ක්‍රමය වේ.

3) `num == 5.`
According to this option, it is false. The `==` operator is used for comparison, not for assignment. This checks if `num` equals `5`, but does not assign the value to the variable.
මෙම විකල්පය අනුව, එය අසත්‍ය වේ. `==` ක්‍රියාකරු භාවිතා කරනුයේ සැසඳීම සඳහා මිස පැවරුම සඳහා නොවේ. මෙය `num` `5`ට සමාන දැයි පරීක්ෂා කරයි, නමුත් විචල්‍යයට අගය ලබා නොදේ.

4) `num : 5.`
According to this option, it is false. This syntax is incorrect in Python. The colon (`:`) is not used to assign values to variables.
මෙම විකල්පය අනුව, එය අසත්‍ය වේ. Python හි මෙම වාක්‍ය ඛණ්ඩය වැරදියි. විචල්‍ය සඳහා අගයන් පැවරීමට colon (`:`) භාවිතා නොවේ.

5) `int num = 5.`
According to this option, it is false. This syntax is more like C-style languages, but in Python, you don't need to specify the type of the variable `(int);` Python is dynamically typed, so simply using `num = 5` is enough.
මෙම විකල්පය අනුව, එය අසත්‍ය වේ. මෙම වාක්‍ය ඛණ්ඩය C-style භාෂා වලට සමාන වේ, නමුත් Python හි, ඔබට විචල්‍යයේ වර්ගය `(int);` සඳහන් කිරීමට අවශ්‍ය නොවේ Python ගතිකව ටයිප් කර ඇත, එබැවින් සරලව `num = 5` භාවිතා කිරීම ප්‍රමාණවත් වේ.

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

10. What will be the output of the given Python code?
දී ඇති පයිතන් කේතයේ ප්‍රතිදානය කුමක් වේවිද?

```
def reverse_string(s):
    reversed_s = s[::-1]
    return reversed_s

text = "Hello, World!"
result = reverse_string(text)
print(result)
```

1) !dlroW ,olleH 2) Hello, World! 3) Hello World!
4) World! Hello, 5) dlroW ,olleH

Answer: 1)

Explanation / පැහැදිලි කිරීම:

The given Python code defines a function `reverse_string(s)` that takes a string `s` as input and returns the reversed version of that string. In the code:
ලබා දී ඇති Python කේතය `s` string එකක් ආදානය ලෙස ගෙන එම string එකේ ප්‍රතිලෝම අනුවාදය ආපසු ලබා දෙන `reverse_string(s)` ශ්‍රිතයක් අර්ථ දක්වයි. කේතය තුළ:

- The string `"Hello, World!"` is passed as the argument to the function.
ශ්‍රිතයට තර්කය ලෙස `"Hello, World!"` string එක සම්මත කර ඇත.

- Inside the function, the string slicing `s[::-1]` reverses the string. The slice `[::-1]` works by starting at the end of the string and moving backwards with a step of `-1`.
ශ්‍රිතය ඇතුළත, string කපා හැරීම `s[::-1]` string ආපසු හරවයි. `[::-1]` පෙත්ත ක්‍රියා කරන්නේ string එකේ කෙළවරින් පටන් ගෙන `-1` පියවරකින් පසුපසට ගමන් කිරීමෙනි.

- The reversed string is stored in `reversed_s`, which is then returned.
ආපසු හරවන ලද string එක `reversed_s` තුළ ගබඩා කර ඇත, එය ආපසු ලබා දෙනු ලැබේ.

- Finally, the reversed string is printed.
අවසාන වශයෙන්, ආපසු හරවන ලද string එක මුද්‍රණය කර ඇත.

So, the output of the code is `!dlroW ,olleH`, which is the reverse of the input string `"Hello, World!"`.
එබැවින්, කේතයේ ප්‍රතිදානය `!dlroW ,olleH` වේ, එය ආදාන string එකේ ප්‍රතිලෝම වන `"Hello, World!"` වේ.