* 1. What is the time complexity of the push() operation in a stack?
     1. **O(1)**
     2. O(log n)
     3. O(n)
     4. O(n log n)
  2. Which of the following data structures uses a Last-In-First-Out (LIFO) order?
     1. Queue
     2. **Stack**
     3. Linked List
     4. Binary Tree
  3. The operation to remove an element from the top of the stack is called:
     1. **pop()**
     2. push()
     3. peek()
     4. remove()
  4. In a stack implemented using an array, what is the maximum number of elements it can hold?
     1. Limited by memory
     2. **Limited by the size of the array**
     3. Limited by the number of elements pushed
     4. Unlimited
  5. Which of the following expressions can be evaluated using a stack-based approach?
     1. **Postfix expression**
     2. Infix expression
     3. Prefix expression
     4. All of the above
  6. Which of the following is a real-world application of a stack data structure?
     1. **Browser history**
     2. Sorting elements
     3. Binary search
     4. Hashing
  7. The minimum number of stacks needed to implement a queue is:
     1. 1
     2. **2**
     3. 3
     4. 4
  8. Consider the following operation performed on a stack of size 5.

Push(1);

Pop();

Push(2);

Push(3);

Pop();

Push(4);

Pop();

Pop();

Push(5);

After the completion of all operation, the number of elements present in stack is?

* + 1. **1**
    2. 2
    3. 3
    4. 4
  1. In a stack-based implementation of an expression evaluator, why are parentheses important?
     1. They indicate the end of the expression.
     2. **They control the order of evaluation.**
     3. They indicate a syntax error.
     4. They are used for multiplication.
  2. The stack used to implement function calls and return addresses in programming languages is known as:
     1. **Call stack**
     2. Memory stack
     3. Data stack
     4. Execution stack
  3. Which operation allows you to access the element at the top of the stack without removing it?
     1. pop()
     2. push()
     3. **peek()**
     4. traverse()
  4. Assume that the operators +, -, × are left associative and ^ is right associative. The order of precedence (from highest to lowest) is ^, x , +, -. The postfix expression corresponding to the infix expression a + b × c - d ^ e ^ f is
     1. **abc × + def ^ ^ -**
     2. abc × + de ^ f ^ -

c) ab + c × d - e ^ f ^

d) - + a × bc ^ ^ def

* 1. In a stack data structure, the element that was pushed onto the stack most recently is called the:
     1. **Top element**
     2. Bottom element
     3. Base element
     4. First element
  2. Which of the following is an example of a stack-based algorithm?
     1. **Depth-first search**
     2. Breadth-first search
     3. Dijkstra's algorithm
     4. Binary search
  3. The average and worst-case time complexity of the pop() operation in a stack is:
     1. **O(1)**
     2. O(log n)
     3. O(n)
     4. O(n log n)
  4. Which data structure is commonly used to implement undo functionality in applications?
     1. Queue
     2. **Stack**
     3. Linked List
     4. Hash Table
  5. A stack-based algorithm is used to evaluate postfix expressions because it eliminates the need for:
     1. Parentheses
     2. Arithmetic operations
     3. **Operator precedence rules**
     4. Operand values
  6. The postfix expression "3 4 + 5 \*" evaluates to:
     1. 27
     2. 23
     3. **35**
     4. 60
  7. Which of the following statements is true regarding a stack?
     1. It is a dynamic data structure.
     2. It follows the FIFO (First-In-First-Out) order.
     3. **It can be implemented using a linked list or an array.**
     4. It is used to implement priority queues.
  8. A stack-based algorithm is used to convert infix expressions to postfix expressions because it helps manage:
     1. Operator associativity
     2. Operand values
     3. **Parentheses**
     4. Operator precedence
  9. The following postfix expression with single digit operands is evaluated using a stack:

8 2 3 ^ / 2 3 \* + 5 1 \* -

Note that ^ is the exponentiation operator. The top two elements of the stack after the first \* is evaluated are:

1. **6, 1**
2. 5, 7
3. 3, 2
4. 1, 5
   1. The expression "((2+3)\*(5-4))/(2+3)" can be evaluated using which stack-based approach?
      1. Postfix evaluation
      2. **Infix evaluation**
      3. Prefix evaluation
      4. Parentheses balancing
   2. Which of the following is a disadvantage of using an array-based stack implementation?
      1. It has a variable size.
      2. **Memory is wasted when the stack is not full.**
      3. Insertions and deletions are slow.
      4. It does not support dynamic memory allocation.
   3. A stack-based algorithm is often used in the implementation of which programming language feature?
      1. **Recursion**
      2. Loops
      3. File I/O
      4. Exception handling
   4. Which of the following is a valid postfix expression for the infix expression "3 + 4 \* (2 - 1)"?
      1. **"3 4 2 1 - \* +"**
      2. "3 4 2 1 - + \*"
      3. "3 4 + 2 1 - \*"
      4. "3 4 + 2 1 \* -"
   5. The stack-based approach is commonly used to solve problems related to which type of graph traversal?
      1. Breadth-first traversal
      2. **Depth-first traversal**
      3. Pre-order traversal
      4. Post-order traversal
   6. The operation to retrieve and remove the element at the top of the stack is called:
      1. **pop()**
      2. push()
      3. peek()
      4. remove()
   7. Which of the following algorithms uses a stack to keep track of the parentheses in an expression?
      1. Depth-first search
      2. Breadth-first search
      3. Dijkstra's algorithm
      4. **Parentheses balancing algorithm**
   8. Following is C like pseudo code of a function that takes a number as an argument, and uses a stack S to do processing.

void fun(int n)

{

Stack S; // Say it creates an empty stack S

while (n > 0)

{

// This line pushes the value of n%2 to stack S

push(&S, n%2);

n = n/2;

}

// Run while Stack S is not empty

while (!isEmpty(&S))

printf("%d ", pop(&S)); // pop an element from S and print it

}

What does the above function do in general?

1. Prints binary representation of n in reverse order
2. **Prints binary representation of n**
3. Prints the value of Logn
4. Prints the value of Logn in reverse order
   1. Following is an incorrect pseudocode for the algorithm which is supposed to determine whether a sequence of parentheses is balanced:

declare a character stack

while ( more input is available)

{

read a character

if ( the character is a '(' )

push it on the stack

else if ( the character is a ')' and the stack is not empty )

pop a character off the stack

else

print "unbalanced" and exit

}

print "balanced"

Which of these unbalanced sequences does the above code think is balanced?

1. **((())**
2. ())(()
3. (()()))
4. (()))()
   1. Consider the following pseudocode that uses a stack. What is output for input "letsfindc"?

declare a stack of characters

while ( there are more characters in the word to read )

{

read a character

push the character on the stack

}

while ( the stack is not empty )

{

pop a character off the stack

write the character to the screen

}

1. letsfindcletsfindc
2. **cdnifstel**
3. letsfindc
4. cdnifstelcdnifstel
   1. Consider the following pseudocode that uses a stack

declare a stack of characters

while ( there are more characters in the word to read )

{

read a character

push the character on the stack

}

while ( the stack is not empty )

{

pop a character off the stack

write the character to the screen

}

What is output for input "abc"?

1. acb
2. bca
3. **cba**
4. abc