SOTI PLACEMENT PROCESS & GUIDELINES

Dr. Vijaya Chandra J

South India Online Test
Date: Saturday, July 27, 2024

Time: 9:30 a.m. - 12:45 p.m.
Duration: 3 hours & 15 min

Prepared by:
Placement & Training Division

Prepared for:
SR University.

Jul 20, 2030 | Study Material

SOTI Coding Test Process

The SOTI Coding Test is a comprehensive assessment designed to evaluate your programming skills through 100% coding questions. You have the flexibility to choose from five programming languages: C++, C#, Java, Python, and JavaScript. The test comprises three coding questions, each accompanied by two to five test cases. These test cases are divided into Default Test Cases, which help ensure that your solution is on the right track but do not carry any marks, and Basic Test Cases, which carry marks and are crucial for maximizing your score. Please attempt ALL the Basic Test Cases. The entire test is allotted a duration of 3 hours and 15 minutes. Ensure you click the submit button before time is up. Failing to submit might result in loss of marks.

Coding Test Process

1. Type of Test: 100% coding questions.

2. Choice of Languages: You will have five languages to choose from, including: C++, C#, Java, Python and JavaScript.

3. # of Questions: 3

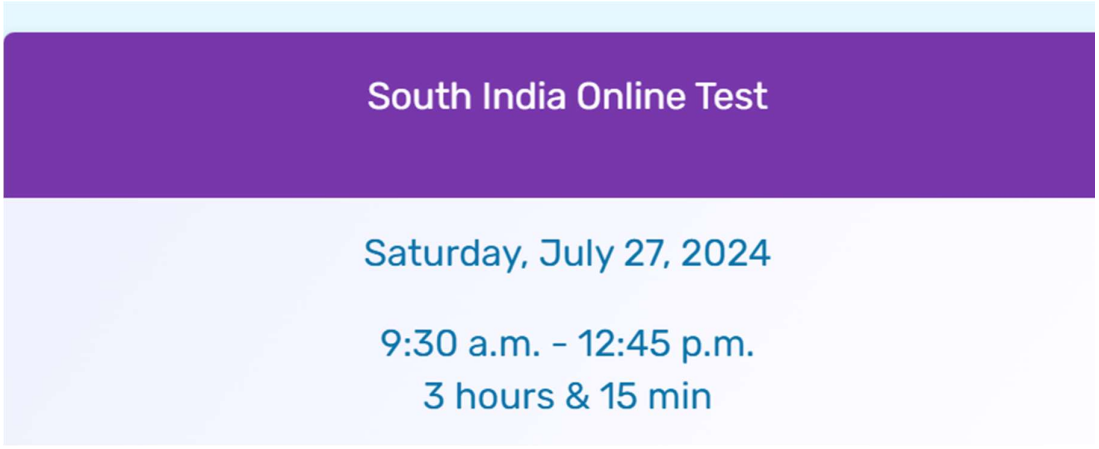
4. # of Test Cases: Each question will have two to five Test Cases. There are two types – Default Test Cases and Basic Test Cases.

a. Default Test Cases will help ensure that your solution is on the right track. These Default Test Cases carry NO marks.

b. Basic Test Cases carry marks. Please attempt ALL the Basic Test Cases.

5. Duration: 3 hours and 15 minutes. Please ensure you click the submit button before time is up. If you do not click submit, you may lose marks.

6. Location: Online. You may take the test on campus from your college or online from home.



SOTI Preparation Tips

To excel in the SOTI coding test, it is essential to follow a structured preparation strategy. Start by practicing coding problems on platforms like LeetCode, HackerRank, or CodeSignal, ensuring you work across all five languages (C++, C#, Java, Python, and JavaScript) to identify your strongest option. Understand the importance of test cases by writing code that successfully passes multiple cases, including edge scenarios. Effective time management is crucial; practice solving problems within a set time limit and learn to allocate your time efficiently among the three questions. Enhance your debugging skills to quickly resolve issues, utilizing print statements or debugging tools available in your chosen language. Focus on language-specific preparation, becoming familiar with syntax, standard libraries, and common data structures and algorithms. Take mock tests to simulate the exam environment, helping you adapt to the pressure and format of the actual test. Always keep an eye on the clock during the exam, aiming to finish coding a few minutes early to review and ensure you submit your work on time. By diligently following these tips and practicing thoroughly, you can significantly improve your chances of success in the SOTI coding test.

Candidate Test Link

1. Candidates will receive the test link by Thursday July 25, 2024 morning in their registered email. They may check the spam/junk folders.

2. If the email Id students entered is wrong, they will not receive the link. There is no need to enquire.

Candidate login process

1. Open the email with the Mettl link and click on it.

2. It will ask for the student data-College name, email etc. Enter the data correctly as given during the registration.

List of things allowed during the test

1. Paper & Pen

2. Calculator

(MOBILE PHONES ARE STRICTLY PROHIBITTED)

Instructions to be given to all candidates

1. DO NOT use mobile phones or other devices while attempting the test.

2. DO NOT use shortcuts (not even Ctrl-C/Ctrl-V/Ctrl-S), toggle between tabs

3. DO NOT open other tabs or browser.

4. DO NOT look elsewhere other than on the screen.

5. DO NOT talk to others.

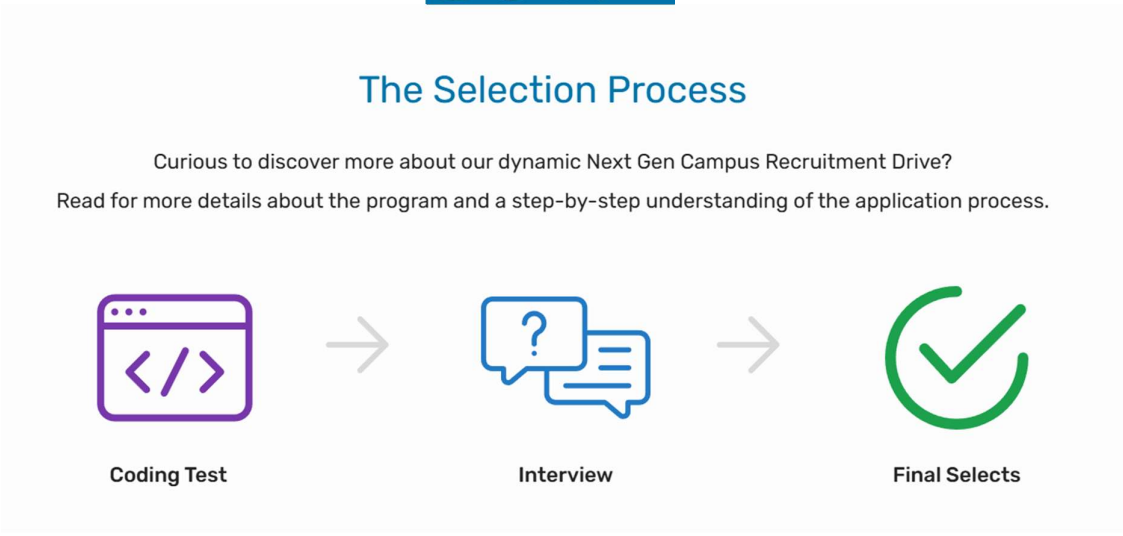
6. The test is proctored (closely watched). After a few attempts, you will be automatically locked out. Once locked out, you will NOT get a second chance to complete the test.

Integrity during the test

1. Please remember at SOTI we highly value integrity and honesty and ZERO tolerance for malpractices.

2. It is a proctored test. Your candidature will be SUMMARILY REJECTED if any flag is raised.

SOTI



Common Question Patterns for Test Preparation

Array manipulation is a fundamental problem pattern often encountered in SOTI exams. It involves performing various operations on arrays to solve problems efficiently. Understanding and mastering this pattern is crucial for achieving high scores in technical interviews and coding assessments. Examples are Find the Sum of Elements in an Array, Rotate an Array to the Left/Right, Find the Maximum Product of Two Integers in an Array.

String manipulation is a pattern involves various operations on strings to solve problems efficiently. These problems require various operations on strings, such as searching, modifying, and analysing string data.

Stack and queue problems are common in coding exams like SOTI, as they test a candidate's understanding of fundamental data structures and their applications. These problems typically involve implementing or utilizing stack and queue operations to solve specific tasks. Example Problems includes, Implement a stack with basic operations such as push, pop, and top. Design a stack that supports push, pop, and retrieving the minimum element in constant time. It will also include different patterns such as Implementing Basic Queue Operations, Validating Parentheses, and Implementing Queue Using Stacks/Stack Using Queues. Mastering stack and queue problems is essential for performing well in SOTI exams. By practicing and understanding these common problem patterns and solutions, you can develop the skills needed to tackle similar challenges efficiently and effectively.

Dynamic Programming (DP) is a powerful technique used to solve problems by breaking them down into simpler sub problems and storing the results of these sub problems to avoid redundant computations. DP is commonly featured in coding exams like SOTI because it tests a candidate's ability to think algorithmically and optimize solutions.

Dynamic Programming (DP) is a critical algorithmic technique used to solve complex problems by breaking them down into simpler overlapping sub problems and storing the results of these sub problems to avoid redundant computations. An overview of common DP problem patterns, including the Fibonacci Sequence, Longest Increasing Subsequence, Knapsack Problem, Edit Distance, and Coin Change Problem. Each pattern demonstrates key concepts and strategies in DP, showcasing how to approach and solve problems efficiently. The Fibonacci Sequence illustrates the fundamental idea of solving a problem recursively while utilizing memorization. The Longest Increasing Subsequence showcases how to build solutions incrementally while maintaining optimal substructure. The Knapsack Problem exemplifies how to make optimal choices to maximize value under constraints. Edit Distance highlights the importance of minimal operations in transforming one sequence into another. Finally, the Coin Change Problem demonstrates finding both the minimum number of coins needed and the number of ways to make change for a given amount. Mastery of these patterns equips individuals with the necessary skills to tackle a wide range of dynamic programming challenges effectively.

Backtracking is a powerful algorithmic technique used for solving problems by incrementally building solutions and abandoning those that fail to meet the required constraints. It systematically explores all potential solutions, often used for combinatorial problems like the N-Queens problem and Sudoku. In these scenarios, backtracking helps to construct solutions step-by-step, backtracking when a partial solution violates constraints, ensuring that only valid solutions are considered.

Graph algorithms focus on the study of graphs and their properties, addressing problems related to connectivity, shortest paths, and cycles within graph structures. These algorithms are essential for tasks such as finding the shortest path between nodes, detecting cycles, and optimizing routes. When combined with backtracking, graph algorithms can efficiently explore and solve complex

problems by navigating through potential solutions and leveraging graph-based techniques to handle large and intricate datasets. Together, these approaches provide robust solutions for a wide range of computational problems, from simple puzzles to complex optimization challenges.