|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Interviewer:** | | Naufal Said | | | **Date of Interview:** | | July 14, 2020 (2 hour) | | |
| **Candidate Name:** | | Bruce Lee | | | **Applied Position:** | | Enterprise Imaging Team, Intermediate Developer (SDE-2) | | |
| **Recommendation:** | | Fit position: , Not fit applied position but may fit others: , Reject | | | | | | | |
| **Overall:** The candidate passed coding test. He has great logical thinking to understand the question, He also familiar with data structure to solve the problem, and he could adjust his design based on certain condition. | | | | | | | | | |
| **Scale:** | **5** – Excellent | | **4** – Strong | **3** – Average | | **2** – Weak | | | **1** – None |
| **Coding Test/Assessment** | | | | | | | | **Rating** | |
| **Q:**  Given one integer, one string,  each digit of integer will be pair with string / reverse-string.  for the example results:   * Input: 98, qux => output “qux9 xuq8”. * Input: 735, bar => output “bar7 rab3 bar5”. * input: 2278, foo => output “foo2 oof2 foo7 oof8”.   Note:  Recommended to splitting the integer not using String object.  **A:**  public static string Task1(int i, string s)         {             var integerList = new List<int>();             while (i > 10)             {                 integerList.Add(i % 10);                 i /= 10;             }             integerList.Add(i);             integerList.Reverse();             var charA = s.ToCharArray();             Array.Reverse(charA);             var reversedS = new string(charA);             var sb = new StringBuilder();             bool isReversed = false;             foreach (var num in integerList)             {                 sb.Append(isReversed ? reversedS : s);                 sb.Append(num);                 sb.Append(" ");                 isReversed = !isReversed;             }              return sb.ToString();         }    **Assessment:**   |  |  |  |  | | --- | --- | --- | --- | | **Items** | **Check** | **Items** | **Check** | | Use of modulo | Ö | Use of StringBuilder | Ö |   **Comments:** 14 Minutes to solved, skilled using debugging tools, prefer use var variable. Good at reading and understanding the question. | | | | | | | | 3 | |
| **Q:**  given n number integers, less than or equal to 1000,  from 1 to n take each integer as input, and apply this rule:   * if input is divided by 3 => print A * if input is divided by 5 => print B * if input is prime number => print P     for example:  n: 15  input: 1 output: “”  input: 2 output: “”  input: 3 output: “AP”  input: 4 output: “”  input: 5 output: “BP”  input: 6 output: “A”  ......  Input: 15 output: “AB”  Note: This function should be handling more condition without modify the core function and make more generic and expendable.  **A:**  public static Dictionary<int, string> Conditions { get; set; } = new Dictionary<int, string>         {             { 3,"A"},             { 5,"B"},             { 7,"C"},             { 10,"D"}         };  public static void Task2(int n)         {             if (n > 1000)             {                 Console.WriteLine("out of range");                 return;             }                while (true)             {                 var input = Console.ReadLine();                 if (!int.TryParse(input, out var number))                 {                     Console.WriteLine("not a number!");                     continue;                 }                    if (number > n)                 {                     Console.WriteLine("out of max limit");                     continue;                 }                    if (number < 1)                 {                     Console.WriteLine("out of min limit");                     continue;                 }                 var sb = new StringBuilder();                 sb.Append($"input: {number} output: ");                    foreach (var condition in Conditions.Where(condition => number % condition.Key == 0))                     sb.Append(condition.Value);                    if (number < 3)                 {                     Console.WriteLine(sb.ToString());                     continue;                 }                    if (IsPrime(number))                     sb.Append("P");                    Console.WriteLine(sb.ToString());             }         }  public static bool IsPrime(int number)         {             for (int i = 2; i < Math.Sqrt(number) + 1; i++)             {                 if (number % i == 0)                     return false;             }                return true;         }  **Assessment:**   |  |  | | --- | --- | | **Items** | **Check** | | Generic/expendable | Ö |   **Comments**: Understand how to modularize the code and familiar to use proper data structure for expendability. | | | | | | | | 4 | |