**Data Structure and Algorithms**

**Assignment 1**

**Submission date 11-11-2022**

|  |
| --- |
| 1. **Mark the following statements as true or false.** 2. The life cycle of software refers to the phases from the point the software was conceived until it is retired.   **Ans:** **True**   1. The three fundamental stages of software are development, use, and discard.   **Ans:** **False**  **These are development, use, and maintenance.**     1. The expression 4n + 2n 2 + 5 is O(n)   **Ans:** **False**  **It’s O(n^2).**   1. The instance variables of a class must be of the same type.   **Ans:** **False.**   1. The function members of a class must be public.   **Ans:** **False**.   1. A class can have more than one constructor.   **Ans:** **True.**   1. A class can have more than one destructor.   **Ans:** **False.**   1. Both constructors and destructors can have parameters.   **Ans: False**  **Only constructors can have parameters, but not destructors.**   1. **Each of the following expressions represents the number of operations for certain algorithms. Characterize the following in terms of Big-O notation?** 2. n^2 + 6n + 4   **Ans: O(n2)**   1. 5n^3 + 2n + 8   **Ans: O(n3)**   1. (n^2 + 1) (3n + 5)   **Ans: O(n3)**   1. 5(6n + 4)   **Ans: O(n)**   1. n + 2log2n – 6   **Ans: O(n)**   1. 4n log2n + 3n + 8   **Ans: O(nlog2n)** |
| 1. **Consider the following function:**   void funcExercise6(int x, int y)  {  int z;  z = x + y;  x = y;  y = z;  z = x;  cout << "x = " << x << ", y = " << y << ", z = " << z << endl;  }  Find the exact number of operations executed by the function  funcExercise6.  **Ans:** The exact number of operations executed by the function is **12.** |
| 1. **Consider the following function:** 2. int funcExercise7(int list[], int size)   2- {  3- int sum = 0;  4- for (int index = 0; index < size; index++)  5- sum = sum + list[index];  6- return sum;  7- }   1. **Find the number of operations executed by the function funcExercise7 if the value of size is 10.**   **Ans: 43.**  Lines 3, 6 have 1 operation each (total: 2).  Lines 4 has a for loop in which index is initialized once so that’s 1 operation as well  Line 4 again has 2 operations in the for loop.  Line 5 has 2 (Total: 4).  Both line 4 and 5 are looped 10 times(with exception of int index)   1. 4(10) + 3 = **43**      1. **Find the number of operations executed by the function funcExercise7 if the value of size is n.**   **Ans: 4n + 3.**  Lines 3, 6 have 1 operation each (total: 2).  Lines 4 has a for loop in which index is initialized once so that’s 1 operation as well  Line 4 again has 2 operations in the for loop.  Line 5 has 2 (Total: 4).  Both line 4 and 5 are looped n times(with exception of int index)   1. **What is the order in terms of Big-O notation of the function funcExercise7?**   **Ans: O(n)** |
| 1. **Characterize the following algorithm in terms of Big-O notation. Also find the exact number of additions executed by the loop. (Assume that all variables are properly declared.)**   for (int i = 1; i <= n; i++)  sum = sum + i \* (i + 1);  **Ans: Big-O notation is:**  n+1=O(n).  **The exact number of additions executed by the loop:**  **2**n Times. |
| 1. **Characterize the following algorithm in terms of Big-O notation. Also find the exact number of additions, subtractions, and multiplications executed by the loop. (Assume that all variables are properly declared.)**   for (int i = 5; i <= 2 \* n; i++)  cout << 2 \* n + i - 1 << endl;  **Ans:**  **2n-5+1 = 2n-4**  The for loop has 2n – 4 iterations. Each time through the loop a fixed number of statements execute. Hence, this algorithm is O(n). Now each time through the loop there is one addition, one subtraction, and one multiplication. Thus, the numbers of additions is 2n – 4, the number of subtractions is 2n – 4, and the number of multiplications is 2n – 4. |
| 1. **Characterize the following algorithm in terms of Big-O notation.**   for (int i = 1; i <= 2 \* n; i++)  for (int j = 1; j <= n; j++)  cout << 2 \* i + j;  cout << endl;  **Ans: O(n2)** |
| 1. **Characterize the following algorithm in terms of Big-O notation.**   for (int i = 1; i <= n; i++)  for (int j = 1; j <= n; j++)/  for (int k = 1; k <= n; k++)  cout << i + j + k; |

**Ans:** ***O(*n3*)*.**

Each *for* loop has *n*number of iterations. The 1st loop has *n* number of

iterations. The 2nd has *n2*. The 3rd has *n3*. Algorithm is ***O(*n3*)*.**