Quiz-1 (A&B): Design and Analysis of Algorithms (Spring-2024)

Question 1 [10 Marks]

Given a string (array of characters, Str[1…n]), complete the recursive function given below **(in pseudocode form)** that returns the reversed string. *Note that you are NOT required to print the reversed string and instead you are required to reverse the string in the memory*. Also, write the asymptotic time complexity of your algorithm. For example, the string "stony" is required to be converted to "ynots" in the memory. Note that this is just an example and your solution should work for a general case Str[1…n].

```
reverse(str, 1, n)    {1st Recursive function call; 1 is the 1st index, n is the
last index, i.e., n is the size of the string}

reverse(str, first, last)
BEGIN
        IF first ≥ last
                Return
        ELSE
                Swap(str[first], str[last])
                reverse(str, first+1, last-1)
        END IF
END
```

Asymptotic Time Complexity:

T(n) = T(n-2) + 1

$\Theta(n)$

Question 2 [5x2=10 Marks]

Provide a worst-case asymptotic time complexity of the following algorithms/ codes by using a suitable asymptotic notation considering a nearest function. Assume that there are no errors/ bugs in the algorithms/ codes. Also, provide the justification for your answer in the last column.

	Algorithm/ code	**Time Complexity**	**Justification**

a)	`for (int i = 1; i<=n*n; i++)`
	`{`
	`      for (int j = i; j <n; j++)`
	`           cout << j % n;`
	`}`
		$\Theta(n^2)$	Outer loop executes $n \times n = n^2$ times.
			Inner loop executes i times, maximum n times only for 1st n iterations of the outer loop.

b)	`ch = 'a';`
	`// Assume that a user enters the`
	`// character 'z' after k attempts`
	`while ( ch != 'z' )`
	`{`
	`      for (int i =n; i >=1; i=i/2)`
	`      {`
	`            cout << ch;`
	`      }`
	`      cin >> ch;`
	`}`
		$\Theta(k\log n)$	While loop executes maximum of k times
			For loop runs from n down to 1 with a decreasing multiplicative factor of 2. So, it executes logn times.

c)	`// Assume that the number of nodes`
	`// in the binary tree is n`
		$\Theta(n)$	In inorder traversal, we traverse each node exactly

	`void printInorder(struct Node* node)`
	`{`
	    `if (node == NULL)`
	        `return;`
	    `printInorder(node->left);`
	    `cout << node->data << "  ";`
	    `printInorder(node->right);`
	`}`
			one time.

d)	`for (int i=1; i<N; i++)`
	`{`
	      `int a = 0, b = 0;`
	      `for (k = 0; k < N; k++)`
	            `a = a + rand();`
	      `for (j = 0; j < M; j++)`
	            `b = b + rand();`
	`}`
		$\Theta(N\times(N+M))$	Outer for loop executes N times.
			Inner for loops execute N+M times.

e)	`Power(a,k)  {calculate a`^{k}`,k≥0}`
	`BEGIN`
	    `IF k = 0 THEN`
	         `Return 1`
	    `ELSE`
	         `Return a × Power(a, k - 1)`
	    `END IF`
	`END`
		$\Theta(k)$	$T(n) = T(n-1) + 1$
			leads to linear complexity.