National University of Computer and Emerging Sciences, Lahore Campus

	Course:	**Data Structures**	**Course Code:**	**CS 2001**
	Program:	**BS (Software Engineering)**	**Semester:**	**Fall 2023**
	Due Date	**22-DEC-2023 at 11:59 pm**	**Total Marks:**	**50**
	Section:	**1A & 1B**	**Page(s):**	**2**
	Type:	**Assignment 6**		

Important Instructions:

1. Put your .doc files into a zip folder and upload it onto the google classroom submission folder. Name your solution file with your roll number, i.e., Assignment6_22L_1111.zip. Assignment in any other format (extension) will not be accepted and will be awarded with zero marks.
2. You are not allowed to copy solutions from other students. We will check your solution for plagiarism using plagiarism checkers. If any sort of cheating is found, negative marks will be given to all students involved.
3. Late submission of your solution is not allowed. For each passing day after deadline, 20% of the marks will be deducted. Three days after the deadline, no submission will be accepted.

Question # 1: [Marks 10]

Huffman encoding is an example of a lossless compression algorithm that works particularly well on text but can, in fact, be applied to any type of file. Using Huffman encoding to compress a file can reduce the storage it requires by a third, half, or even more, in some situations. You'll be impressed with the compression algorithm.

"The definition of heap is that it is a complete binary tree that conforms to the heap order"

Compress the given sentence using Huffman encoding algorithm. You have to show complete steps like table, tree etc. After generating codes replace each character with their corresponding code.

Question # 2: [Marks 5]

Heapify the elements of the following array (reading from left to right) into a Min Heap and show that Min Heap contents in the form of array (as shown below) and tree.

Original Array	6	5	3	9	1	2	10	8	-

Question # 3 [Marks 10]

623+-382/+*2↑3+

Convert the given postfix expression into an equivalent expression tree with the help of a stack. You have to show the complete steps.

Question # 4: [Marks 15]

Insert the following strings into the two different hash tables:

You can use the following hash function.

```
int hashCode(char* s){
int sum=0;
for(int i=0; i<strlen(s);i++){
   sum = sum+s[i];
}
return sum % TableSize;
}
```

You should insert the objects from left (starting with 'Dijkstra') to right. For each bucket, indicate the total number collisions experienced when inserting the item(s) in the space to the left.

a. A separate chaining hash table with 10 buckets. Insert at the **tail** of each linked list.
b. An open addressing hash table using linear probing with 17 buckets.

Question # 5: [marks 10]

Apply BFS on the graph given below and show the resultant tree.