HERITAGE INSTITUTE OF TECHNOLOGY
LAB ASSIGNMENTS
Algorithm Implementation Lab (CSEN 2211)
B.Tech (CSE) 2nd Year 2nd Semester
Session: 2017 – 2018

Day 1 a) WAP to solve Towers-of-Hanoi problem using Recursive procedure

b) Write two recursive programs to compute X^n, where both X & n are integers, one computes it in O(n) time & the other in O(log n) time.

Day 2 a) WAP to implement QUICK sort using Divide-and-Conquer strategy. Estimate the number of comparisons for different pivots where the input size is 10,000, 1,00,000 and 10,00,000.

b) Implement randomized Quick sort and compare its performance with the previous algorithm by testing average case running time. You should write a separate sub program to generate random permutation for 10 numbers (0 - 9), that has to be used as a sub-part of the complete program.

Day 3 WAP to implement Heap Sort algorithm. Run your program for five randomly generated numbers of size 1,000 and then find average number of comparisons.

Day 4 WAP to multiply a chain of matrices optimally using iterative version of Dynamic Programming approach. Check your program for the sequence of 4 matrices <M1, M2, M3, M4> whose sequence of dimensions is <13, 5, 89, 3, 34>.

Day 5 WAP to find the longest-common-subsequence (LCS), as well as the length of the LCS of two given sequences X & Y using Dynamic Programing Approach. Your program may run on the two given sequences X = <A, B, C, B, D, A, B> and Y = <B, D, C, A, B, A>

Day 6 WAP to implement Djikstra's Algorithm for Single Source Shortest Path problem. Your program should use priority queue implementation. Check your program on the directed graph represented by the following length matrix:

	s	t	x	y	z
s	0	10	∞	5	∞
t	∞	0	1	2	∞
x	∞	∞	0	∞	4
y	∞	3	9	0	2
z	7	∞	6	∞	0

Day 7 a) WAP to solve fractional Knapsack problem using Greedy strategy. Observe the solution given by your program by considering the following greedy strategy:

i) fill up the Knapsack by taking objects which has the highest profit per unit weight

Check your program for the data set: N=3, M=20, <p_1, p_2, p_3>=<25, 24, 15>, <w_1, w_2, w_3>=<18, 15, 10>

b) WAP to solve 0/1 Knapsack problem using dynamic programming approach. Check your program for the sample dataset like, N = 3, M = 30, <p_1, p_2, p_3> = <50, 60, 140>, <w_1, w_2, w_3> = <5, 10, 20>

Day 8 WAP to find the MST of a given graph using Prim's algorithm. Check your program on a graph whose length matrix is given as follows. Your program should use priority queue implementation.

	a	b	c	d	e	f	g	h	i
a	0	4	∞	∞	∞	∞	∞	8	∞
b	4	0	8	∞	∞	∞	∞	11	∞
c	∞	8	0	7	∞	4	∞	∞	2
d	∞	∞	7	0	9	14	∞	∞	∞
e	∞	∞	∞	9	0	10	∞	∞	∞
f	∞	∞	4	14	10	0	2	∞	∞
g	∞	∞	∞	∞	∞	2	0	1	6
h	8	11	∞	∞	∞	∞	1	0	7
i	∞	∞	2	∞	∞	∞	6	7	0

Day 9 WAP to find the MST of a given graph using Kruskal's algorithm. Check your program on a graph whose length matrix is given above. Your program should use UNION & FIND operations on Disjoint data structure.

Day 10 a) WAP to implement Topological Sorting of a DAG. Your program should check whether the given graph is DAG or not. Check your program on a directed graph which is given as below:

	A	B	C	D	E	F	G	H
A	0	∞	∞	1	1	∞	∞	∞
B	∞	0	∞	∞	1	∞	∞	∞
C	1	1	0	∞	∞	1	1	∞
D	∞	∞	∞	0	1	1	∞	∞
E	∞	∞	∞	∞	0	1	∞	∞
F	∞	∞	∞	∞	∞	0	∞	1
G	∞	∞	∞	∞	∞	∞	0	∞
H	∞	∞	∞	∞	∞	∞	∞	0

b) WAP to find strongly connected components in a graph. Check your program on a directed graph which is given as below:

	A	B	C	D	E	F	G	H
A	0	1	∞	∞	∞	∞	∞	∞
B	∞	0	1	∞	1	1	∞	∞
C	∞	∞	0	1	∞	∞	1	∞
D	∞	∞	1	0	∞	∞	∞	1
E	1	∞	∞	∞	0	1	∞	∞
F	∞	∞	∞	∞	∞	0	1	∞
G	∞	∞	∞	∞	∞	1	0	1
H	∞	∞	∞	∞	∞	∞	∞	1

Day 11 WAP to implement Knuth-Morris-Pratt (KMP) algorithm for pattern matching. Check your program for the following Text & Pattern:

T: b a c b a b a b a a b c b a b

P: a b a b a c a

Day 12 WAP to implement Ford-Fulkerson algorithm to get maximum flow of a given flow network. Check your program on a directed graph whose length matrix is given below, where s denotes source and t denotes destination:

	s	u	v	t
s	0	20	10	∞
u	∞	0	30	10
v	∞	∞	0	20
t	∞	∞	∞	0

*****Note: Run all the programs using C language on Linux Platform. Estimate the performance of all the programs in the cases, like best, worst & average. For all the graph algorithms use adjacency list representation of all the graphs. For all experiments (except Day 1) input should be taken from files.**