संगणक विज्ञान एवं अभियांत्रिकी विभाग
Department of Computer Science & Engineering
National Institute of Technology Sikkim, Ravangla, South Sikkim-737139
(AN INSTITUTE OF NATIONAL IMPORTANCE, MINISTRY OF EDUCATION (MoE), GOVT. OF INDIA)

List of Laboratory Programs

Subject: Advanced Data Structures and Algorithms Laboratory (CS21201)
Semester: 1st Semester, M.Tech (AI and ML)

Programs:

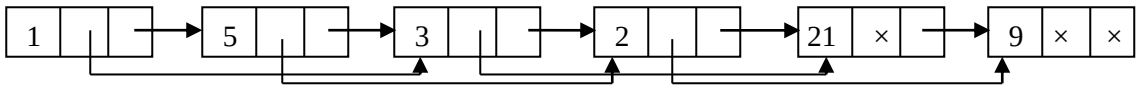
Prog 1. Design a function DeleteAll(struct LIST *PList, int x) to delete all the repeated item *x* from a linked-list.
Input List: 6 → 7 → 3 → 4 → 3 → 1 → 3 → 2
Output List (for *x* = 3): 6 → 7 → 4 → 1 → 2

Prog 2. Design a function DeleteRepeated(struct LIST *PList, int x) to delete all the repeated item *x* from a linked-list but keep *x* once only in its first occurrence.
Input List: 6 → 7 → 3 → 4 → 3 → 1 → 3 → 2
Output List (for *x* = 3): 6 → 7 → 3 → 4 → 1 → 2

Prog 3. Write a function to reverse the list by changing the links of the blocks.
Input List: 6 → 7 → 3 → 4 → 2 → 1
Output List: 1 → 2 → 4 → 3 → 7 → 6

Prog 4. (a) Design a function to add two polynomials. The function returns the resultant polynomial.
(b) Design a function to multiply two polynomials. The function returns the resultant polynomial.

Prog 5. Assume a linked list as given in the following figure (consider the figure just as an example).



Write program with following methods:
(a) A method to insert a new element after a given element in the linked list.
(b) A method to insert a new element at the end of the linked list.
(c) A method to print all the elements of the list.

Prog 6. (a) Design a function to delete the element after a given element from a Doubly Linked List.
(b) Design a function to delete the element before a given element from a Doubly Linked List.

Prog 7. Write program with following methods on Binary Search Tree (BST):
(a) A method to insert a new element.
(b) A method to search a given number.
(c) A method to delete a given number.
(d) In-order, Pre-order and Post-order traversal.
(e) A method to find out height of the tree.
(f) A method to find out total number of nodes.
(g) A method to find out total number of internal nodes.
(h) A method to find out total number of external nodes.
(i) A method to find out total number of leaf nodes.

Prog 8. Assume a series of positive integer numbers, S = {1, 2, 3, 4, 5, 6, 8, 9, 10, 12, …}. This is the series of all the positive integer numbers whose prime factors only include 2, 3, and 5. Note that 1 is typically treated as the first element of this series. The 10th element is 12, because first 10 numbers are 1, 2, 3, 4, 5, 6, 8, 9, 10, 12. Write a program to find the *n*-th element of this series. In this program, input is the *n* as positive integer.

***Hints*:** Use MinHeap. We will keep a min heap and initially insert 1 as the first number. The heap is {1}.

- Now, for next numbers, we ExtractMin from heap (i.e., root 1) and insert the multiples of 2, 3, and 5. Now, the heap becomes {2, 3, 5}.
- For next number, we extract min from heap which is 2. Now, we insert 2×2=4, 2×3=6, 2×5=10 into the heap and the heap becomes {3, 4, 5, 6, 10}.
- Next, we extract 3 and push 3×2=6, 3×3=9, and 3×5=15. Note that, we have already inserted 6, so we should not push another 6 into heap. Now heap becomes {4, 5, 6, 9, 10, 15}.
- We extract one element at each step until we extract *n*-th element.

Prog 9. We are given N objects and a knapsack or bag. The object i has a weight w_i and the maximum capacity of the knapsack is M. If a fraction x_i of the object i is placed in the bag then a profit $p_i x_i$ is earned and a weight $w_i x_i$ is filled up in the bag. The objective is to obtain a filling of the knapsack that maximizes the profit. Given N objects with $\{w_1, w_2, \ldots, w_N\}$ and $\{x_1, x_2, \ldots, x_N\}$, write a ***Greedy Algorithm*** to solve the Knapsack Problem.

Prog 10. Given a sequence $P = \{p_0, p_1, \ldots, p_N\}$ be the set of integers, write a ***divide and conquer algorithm*** to calculate $m[i, j]$ as per the following recursive equation. Write a function *CalculateM*(*i, j*) with integer return type. Analyze time complexity of the algorithm.

$$m[i,j] = \begin{cases} 0, & \text{If } i = j \\ \min_{i \le k < j}\{m[i,k] + m[k+1,j] + p_{i-1}p_k p_j\}, & \text{If } i < j \end{cases}$$

Prog 11. For the above Prog 10, write a ***dynamic programming algorithm*** to calculate $m[i, j]$. Write a function *CalculateMDynamic*(*i, j*) with integer return type. Analyze time complexity of the algorithm.

Prog 12. Given two sequences $X = \{x_1, x_2, \ldots, x_N\}$ and $Y = \{x_1, y_2, \ldots, y_M\}$ be the sequence of characters, write a ***divide and conquer algorithm*** to calculate $c[i, j]$ as per the following recursive equation. Write a function *CalculateC*(*i, j*) with integer return type. Analyze time complexity of the algorithm.

$$c[i,j] = \begin{cases} 0, & \text{If } i = 0 \text{ or } j = 0 \\ c[i-1, j-1] + 1, & \text{If } i, j > 0 \text{ and } x_i = y_j \\ \max\{c[i, j-1], c[i-1, j]\}, & \text{If } i, j > 0 \text{ and } x_i \neq y_j \end{cases}$$

Prog 13. For the above Prog 12, write a ***dynamic programming algorithm*** to calculate $c[i, j]$. Write a function *CalculateCDynamic*(*i, j*) with integer return type. Analyze time complexity of the algorithm.