# National Institute of Technology Calicut
# Department of Computer Science and Engineering
## B. Tech. (CSE) – Third Semester
## CS2092D: Programming Laboratory
## Extra Question

**1.** Write a menu driven program to implement queue using linked list. The input should be read from the file ***input.txt*** and output should be written to the file ***output.txt.*** Your program should include the following functions.

| | |
|---|---|
| **enqueue( Q, element)** – | Inserts the data specified by element into queue Q. |
| **dequeue(Q)** – | Removes and returns the first-inserted element of the queue if the queue is not empty, otherwise print 'EMPTY'. |
| **print(Q) -** | Display all the elements of Q in the order of their insertion if the queue is not empty, otherwise print 'EMPTY'. |

**Input File Format**

a) The input consists of multiple lines, each line of the input contains a character from {'e', 'd', 'p', 's'} followed by zero or one integer. The integer, if given, is in the range 0 to $2^{31}$ .

b) The character 'e' means enqueue the next integer from the input into the queue. In this case, the next integer (>= 0) is given on the same line as the character 'e', separated by a space.

c) The character 'd' means dequeue and output element from the front of the queue. Output “-1”, if the queue was originally empty.

d) The character 'p' means show all elements in the queue. In this case, output all elements of the queue on a single line, separated by space, starting with the element at the front. Output “-1”, if the queue was originally empty

e) The character 's' means “stop the program”

**Output File Format:**
The output (if any) of each command should be printed on a separate line.

**Example:**

**Sample Input:**

***input.txt***
e 2
e 5
e 7
d

p
e 10
d
p
d
d
d
p
e 3
p
s

**Sample Output:**

***output.txt***

2
5 7
5
7 10
7
10
-1
-1
3

**2.** Write a menu driven program to implement a binary tree using array and perform inorder traversal on it. You may declare a global array *T* and perform all operations on it. Your program must implement the following functions:

**main()** - repeatedly reads a character c', 'p', 'a' or 'i' from the terminal and calls the given functions appropriately until character 's' is entered.

**create_tree()**- reads a binary tree as an array of integers from the terminal and stores it in the global binary tree *T*. Use -1 to represent null node.

**print( r)** - prints the binary tree in the array order starting from the root r. Use -1 to represent null node

**inorder( r)** - recursive function to print the inorder traversal of binary tree T rooted at r.

**preorder_add( r, k )**- recursive function that increments the key value of each node in the binary tree rooted at r by the constant k by traversing in preorder, where 1<=k<=10.

**INPUT FORMAT**

a) The input consists of multiple lines. Each line contains a character from {'c','p', 'a','i', 's'} followed by zero or n integers. Assume 1 <= n <= 20

b) **Character 'c':** character 'c' specifies create tree with 'n' nodes.

c) The next line contains ‘n’ space separated integers which are the elements of the array.

d) Consider the binary tree from the input array as follows .
The root node is stored as the first element of the array, tree [0]. For every index i of the array, the node tree [i]’s left child is stored at tree [2i + 1] & right child at tree[ 2i + 2]. If a node does not have a child, -1 is stored in the corresponding index.

e) **Character 'p':** prints the binary tree ***T*** in its array order.

f) **Character 'i':** prints the inorder traversal of the binary tree.

g) **Character 'a':** character ‘a’ is followed by a constant k, where 1<=k<=10.
Increments the key value of each node in the binary tree rooted at ***r*** by the constant ***k*** by traversing in preorder.

h) **Character 's':** “terminates the program”.

**Output format:**

The output (if any) of each command should be printed on a separate line

**Sample Input:**

c 7
2 3 4 -1 -1 -1 6
p
i
a 4
p
s

**Sample Output:**

2 3 4 -1 -1 -1 6
3 2 4 6
6 7 8 -1 -1 -1 10

**3.** Write a menu driven program to implement a binary search tree (BST) using linked list. Your program must implement the following functions:

**main()** - repeatedly reads a character 'c', 'p’, 'i', 'po', 'a' or 'f' from the terminal and calls the given functions appropriately until character 's' is entered.

**create_tree()**- reads the elements of the binary search tree from the terminal and create the binary search tree T.

**preorder( r)** - recursive function to print the preorder traversal of the binary search tree T rooted at r.

**inorder( r)** - recursive function to print the inorder traversal of the binary search tree T

rooted at r.

**postorder( r )-** recursive function to print the inorder traversal of the binary search tree *T* rooted at *r*.

**add_node( T,k ) -** adds a new node with value *k* to the binary search tree *T* .

**find(T, k) -** search for a node with value k in T. If k is present in T, print the inorder predecessor of the node with value k . Otherwise print “NOT FOUND”. If the node with value k is root node print “ROOT NODE”.

**Input Format**

a) The input consists of multiple lines. Each line contains a character from {'c', 'p’, 'i', 'po', 'a' or 'f'} followed by zero or n integers. Assume 1 <= n <= 20

b) **Character 'c':** character ‘c’ specidies create tree with ‘n’ nodes. The next line contains ‘n’ space separated integers which are the elements of the BST. Create an initial BST with the ‘n’ elements.

c) **Character 'p':** prints the preorder traversal of the BST.

d) **Character 'i':** prints the inorder traversal of the BST.

e) **Character 'po'**: prints the postorder traversal of the BST.

f) **Character 'a':** character ‘a’ is followed by an integer k. Inserts a new node with value *k* to the BST.

g) **Character 'f'**: character ‘f’ is followed by an integer k. Search for a node with value k in BST

h) **Character 's':** “terminates the program”.

**Output format:**

The output (if any) of each command should be printed on a separate line

**Sample Input:**

```
c 7
22 18 20 35 10 19 25
p
i
po
a 5
i
f 19
f 6
s
```

**Sample Output:**

22 18 10 20 19 35 25
10 18 19 20 22 25 35
10 19 20 18 25 35 22
5 10 18 19 20 22 25 35
18
NOT FOUND