University of Sargodha

BS 3rd Term Examination 2016

Subject: I. T Paper: Data Structure & Algorithm (CMP-3112)

Time Allowed: 2:30 Hours Maximum Marks: 80

Objective Part (Compulsory)

Q.No.1. Write short answers of the following questions in 2-4 lines only. (16x2=32)

1. What is a linked-list?
2. What is stack?
3. What operations can be performed on Queues?
4. Why we need to do algorithm analysis?
5. What is binary search?
6. What is selection sort?
7. What is a graph?
8. What is a tree?
9. What is shell sort?
10. How breadth first traversal works?
11. What is a heap in data structure?
12. What is a recursive function?
13. What is tower of hanoi?
14. What is fibonacci series?
15. What is adjacency list?
16. What is hashing?

Subjective Part

Note: Attempt any four questions. **(4x12=48)**

Q.No.2. The following algorithm is supposed to compute the product of the elements of its input array. Prove that the algorithm is correct.

```
Input: Array A[1,..,n], n >= 1
Output: Product of the array's elements

PROD(A)
1.   tulo = 1
2.   i = 1
3.   while i<=n
4.       tulo = tulo*A[i]
5.       i = i+1
6.   return tulo
```

Q.No.3. Depict binary trees with heights 2, 3, 4, 5, and 6. All should be depicted with the following seven keys 1,4,5,10,16,17, and 21.

Q.No.4. Given an array A={12, 11, 13, 5, 6}. Sort it out using a technique illustrated in insertion sort. You have to discuss only the passes in detail and there is no need to write an algorithm of insertion sort.

Q.No.5. Algorithms A and B sort their input arrays. Algorithm A performs $32 \times n\lg(n)$ operations and algorithm B performs $3 \times n^2$ operations, when the array is of size n. Figure out, when to use algorithm A and when to use algorithm B if the size of the array is known.

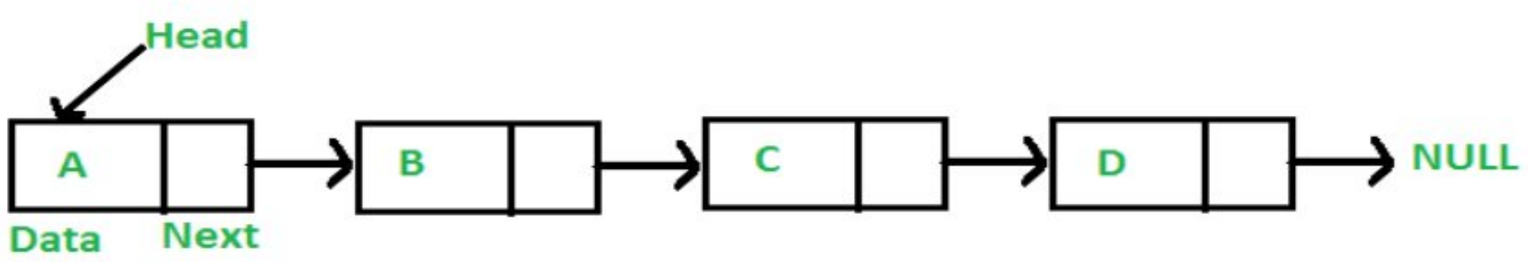
Q.No.6. Write a program to insert or delete item from a circular queue.

OBJECTIVE

Q 1: Short Questions Answers

i. What is a inked-list?

Like arrays, Linked List is a linear data structure. Unlike arrays, linked list elements are not stored at contiguous location; the elements are linked using pointers.



Arrays can be used to store linear data of similar types, but arrays have following limitations.
1) The size of the arrays is fixed: So we must know the upper limit on the number of elements in advance. Also, generally, the allocated memory is equal to the upper limit irrespective of the usage.
2) Inserting a new element in an array of elements is expensive, because room has to be created for the new elements and to create room existing elements have to shifted.

Advantages of Linked lists over arrays

1) Dynamic size

2) Ease of insertion/deletion

ii. What is Stack?

Stack is an ordered list of similar **data** type. **Stack** is a **LIFO(Last in First out)structure** or we can say FILO(First in Last out). **push()** function is used to insert new elements into the **Stack** and **pop()** function is used to remove an element from the **stack**.

iii. What operations can be performed on Queues?

- **Traversal**- This is the process of accessing each and every element of the queue.
- **Enque-** This is an operation where you can add an element to the end of the queue.
- **Deque-** This is an operation where you can delete an element from the starting of the queue.
- **Searching-** You can search through the queue for an element of your choice and display it's position in the queue.
- **Merging-** You can merge two queues together, one behind the other.

iv. Why we need to do algorithm analysis?

Efficiency of an algorithm can be analyzed at two different stages, before implementation and after implementation. They are the following −

A Priori Analysis This is a theoretical analysis of an algorithm. Efficiency of an algorithm is measured by assuming that all other factors, for example, processor speed, are constant and have no effect on the implementation.

A Posterior Analysis − This is an empirical analysis of an algorithm. The selected algorithm is implemented using programming language. This is then executed on target computer machine. In this analysis, actual statistics like running time and space required, are collected.

We shall learn about a priori algorithm analysis. Algorithm analysis deals with the execution or running time of various operations involved. The running time of an operation can be defined as the number of computer instructions executed per operation.

v. What is binary search?

Given a sorted array arr[] of n elements, write a function to search a given element x in arr[]. A simple approach is to do **search. The** time complexity of above algorithm is O(n). Another approach to perform the same task is using Binary Search.

Binary Search: Search a sorted array by repeatedly dividing the search interval in half. Begin with an interval covering the whole array. If the value of the search key is less than the item in the middle of the interval, narrow the interval to the lower half. Otherwise narrow it to the upper half. Repeatedly check until the value is found or the interval is empty.

vi. What is selection sort?

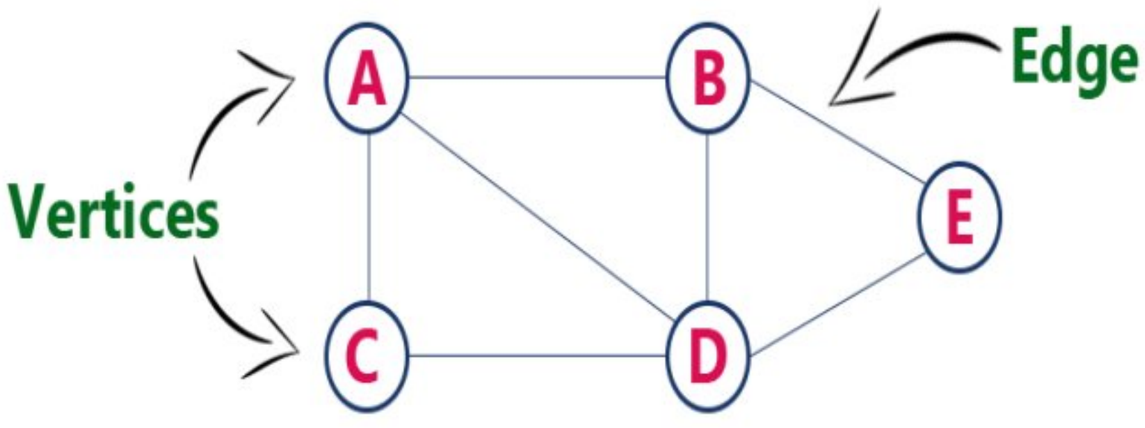
The list is divided into two sub lists, sorted and unsorted, which are divided by an imaginary wall. We find the smallest element from the unsorted sub list and swap it with the element at the beginning of the unsorted data.

After each selection and swapping, the imaginary wall between the two sub lists move one element ahead, increasing the number of sorted elements and decreasing the number of unsorted ones.

Each time we move one element from the unsorted **sub list** to the sorted **sub list**, we say that we have completed a sort pass (round). A list of ***n*** elements requires ***n-1*** passes to completely rearrange the data.

vii. What is a graph?

A **graph** is a pictorial representation of a set of objects where some pairs of objects are connected by links. The interconnected objects are represented by points termed as vertices, and the links that connect the vertices are called edges.

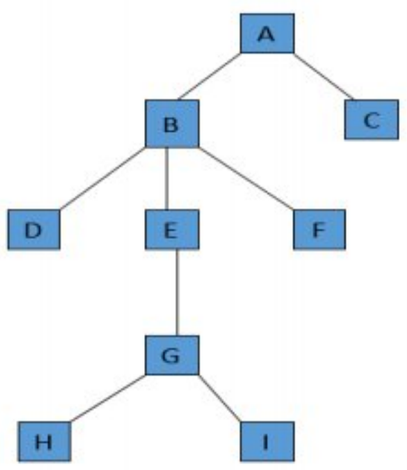


viii. What is a Tree?

Trees are non-linear data structures. Trees are mainly used to represent data containing a **hierarchical relationship** between elements, for example, records, family trees and table of contents. Consider a parent-child relationship.

A tree is an abstract model of a hierarchical structure that consists of nodes with a parent-child relationship. Tree is a sequence of nodes. There is a starting node known as a root node. Every node other than the root has a parent node. Nodes may have any number of children.

- Tree



ix. What is Shell Sort?

Shell sort is a highly efficient sorting algorithm and is based on insertion sort algorithm. This algorithm avoids large shifts as in case of insertion sort, if the smaller value is to the far right and has to be moved to the far left.

This algorithm uses insertion sort on a widely spread elements, first to sort them and then sorts the less widely spaced elements. This spacing is termed as interval.

Running time analysis:

Best case: O(N logn)

Worst case: O(N2)

x. How breadth first traversal works?

Breadth First search (**BFS**) is an algorithm for traversing or searching tree or graph data structures. It starts at the tree root (or some arbitrary node of a graph, sometimes referred to as a "**search** key"') and explores the neighbor nodes **first**, before moving to the next level neighbors. The level-by-level traversal is called a ***breadth-first***

traversal because we explore the *breadth*, i.e., full width of the tree at a given level, before going *deeper*.

xi. What is a heap in data structures?

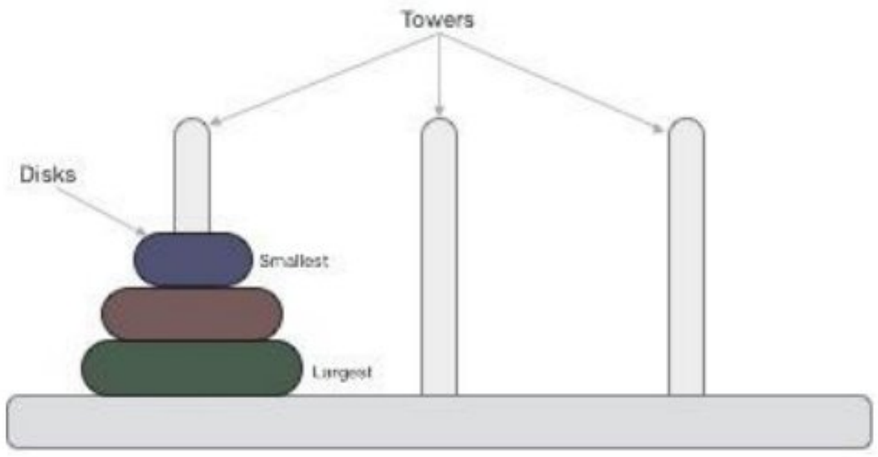
In computer science, a heap is a specialized tree-based data structure that satisfies the heap property: if P is a parent node of C, then the key (the value) of P is either greater than or equal to (in a ***max heap***) or less than or equal to (in a ***min heap***) the key of C. The node at the "***top***" of the heap (with no parents) is called the ***root*** node.

xii. What is a recursive function?

A **recursive function** (DEF) is a **function** which either calls itself or is in a potential cycle of **function** calls. As the definition specifies, there are two types of **recursive functions**. Consider a **function** which calls itself: we call this type of **recursion** immediate **recursion.**

xiii. What is tower of Hanoi?

Tower of Hanoi, is a mathematical puzzle which consists of three towers (pegs) and more than one rings is as depicted. These rings are of different sizes and stacked upon in an ascending order, i.e. the smaller one sits over the larger one. There are other variations of the puzzle where the number of disks increase, but the tower count remains the same.



The mission is to move all the disks to some another tower without violating the sequence of arrangement. A few rules to be followed for Tower of Hanoi are

- Only one disk can be moved among the towers at any given time.
- Only the "top" disk can be removed.
- No large disk can sit over a small disk.

xiv. What is Fibonacci series?

Fibonacci series generates the subsequent number by adding two previous numbers.
Fibonacci series starts from two numbers − **F0 & F1**. The initial values of **F0 & F1** can be taken **0, 1** or **1, 1** respectively.
Fibonacci series satisfies the following conditions −

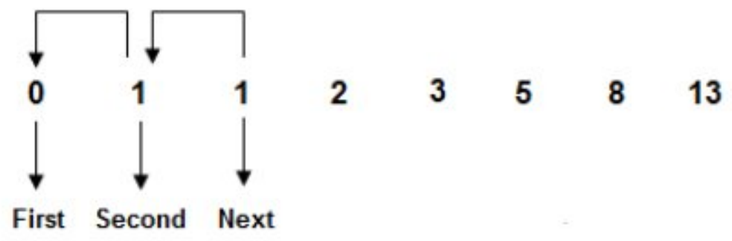
Fn = Fn-1 + Fn-2

Hence, a Fibonacci series can look like this −

F8 = 0 1 1 2 3 5 8 13

or, this −

F8 = 1 1 2 3 5 8 13 21




```
next = first + second;

first = second;

second = next;
```

xv. What is adjacency list?

Graph is a data structure that consists of following two components:

1. A finite set of vertices also called as nodes.

2. A finite set of ordered pair of the form (u, v) called as edge. The pair is ordered because (u, v) is not same as (v, u) in case of a directed graph (di-graph). The pair of the form (u, v) indicates that there is an edge from vertex u to vertex v. The edges may contain weight/value/cost.

Following two are the most commonly used representations of a graph.

1. Adjacency Matrix

2. Adjacency List

Adjacency List:

An array of linked lists is used. Size of the array is equal to the number of vertices. Let the array be array[]. An entry array[i] represents the linked list of vertices adjacent to the ***i***th vertex. This representation can also be used to represent a weighted graph. The weights of edges can be stored in nodes of linked lists.

xvi. What is hashing?

In computing, a hash table (hash map) is a data structure which implements an associative array abstract data type, a structure that can map keys to values. A hash table uses a hash function to compute an index into an array of buckets or slots, from which the desired value can be found.

A technique used to uniquely identify a specific object from a group of similar objects. To convert a range of key values into a range of indexes of an array by using a **hash function.**

SUBJECTIVE

Q 2:

Q.No.2. The following algorithm is supposed to compute the product of the elements of its input array. Prove that the algorithm is correct.

```
Input: Array A[1,..,n], n >= 1
Output: Product of the array's elements

PROD(A)
1.   tulo = 1
2.   i = 1
3.   while i<=n
4.       tulo = tulo*A[i]
5.       i = i+1
6.   return tulo
```

- **Total Correctness of an Algorithm** depends on the following conditions.
  i. Correct input data is the data which satisfies the initial condition of the speciation.
  ii. Correct output data is the data which satisfies the final condition of the speciation.

✓ **Dentition**

An algorithm is called **totally correct** for the given speciation if and only if for any correct input data it:

a) stops and
b) returns correct output

- **Partial Correctness of Algorithm**

Usually, while checking the correctness of an algorithm it is easier to separately:

i. First check whether the algorithm stops
ii. then checking the "remaining part". This "remaining part" of correctness is called **Partial Correctness** of algorithm.

✓ **Dentition**

An algorithm is partially correct if satisfies the following condition: If the algorithm receiving correct input data stops then its result is correct.

❖ The Given Algorithm is correct as it satisfied all the stages/steps of Algorithms.
- Variables are declared and initialized
- The condition **While i<=n** justify the working to find the output i.e Product of Array's elements.
- The formula is correct according to the condition

Tulo=tulo*A[i]

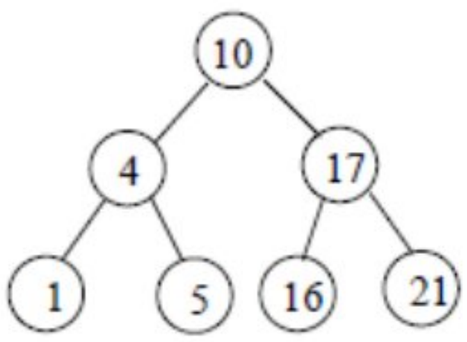
Here A[i] give the elements of Array names **A**, and the next statement i=i+ 1 provides the increment in the array's indexes.

and loop will run **n** times. And at the end it'll return **tulo** the final product of array's elements.

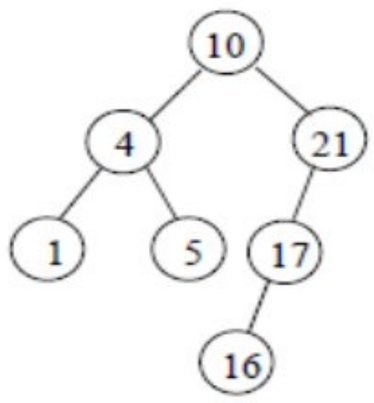
Q 3: Depict binary tree with heights 2,3,4,5 and 6. All should be depicted with the following seven keys 1, 4,5,10,16,17,21.

For Key values 1, 4,5,10,16,17,21 the binary trees of heights 2,3,4,5 and 6 are following

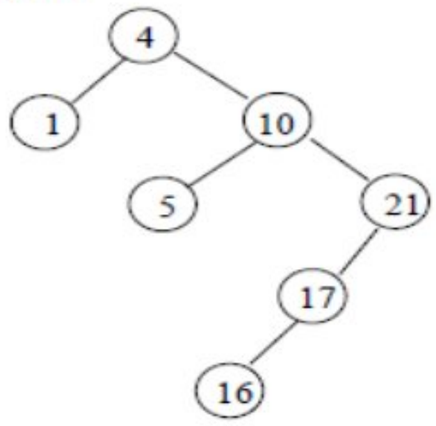
Height 2



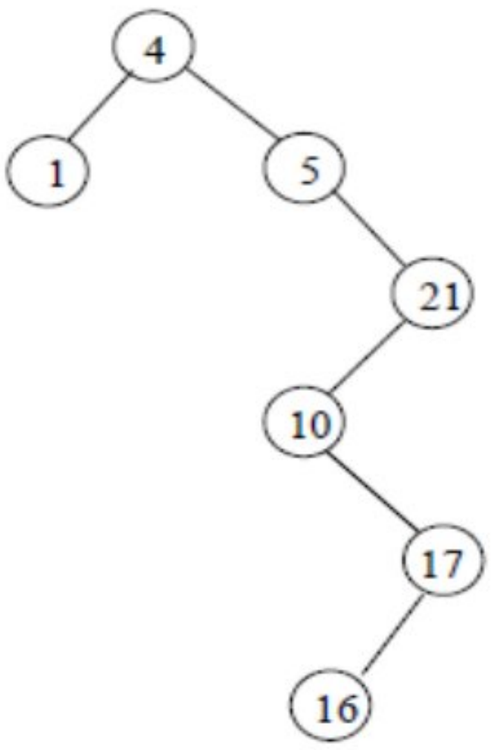
Height 3



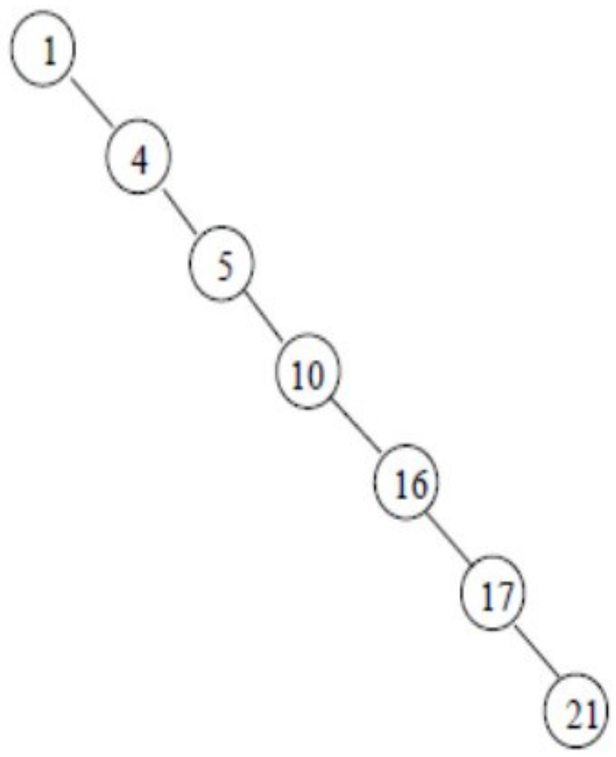
Height 4



height 5



Height 6



Q 4: Given an Array A= {12, 11, 13, 5, 6}. Sort it out using a technique illustrated in insertion sort. You have to discuss only the passes in details and there is no need to write an algorithm of insertion sort.

In insertion sort we have two group of items:

i. sorted group, and
ii. unsorted group

Initially, all items in the unsorted group and the sorted group is empty. We assume that items in the unsorted group unsorted. We have to keep items in the sorted group sorted. Pick any item from unsorted, then insert the item at the right position in the sorted group to maintain sorted property. Repeat the process until the unsorted group becomes empty.

Original Array: {12, 11, 13, 5, 6}

After Pass 1: {11, 12, 13, 5, 6}

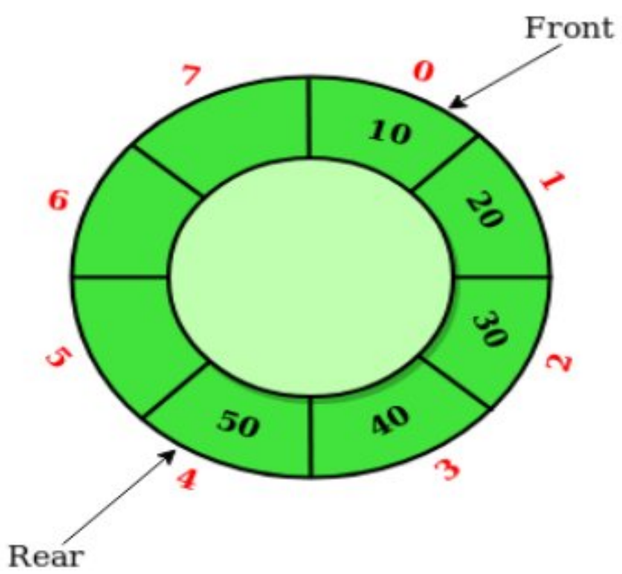
After Pass 2: {11, 12, 13, 5, 6}

After Pass 3: {5, 11, 12, 13, 6}

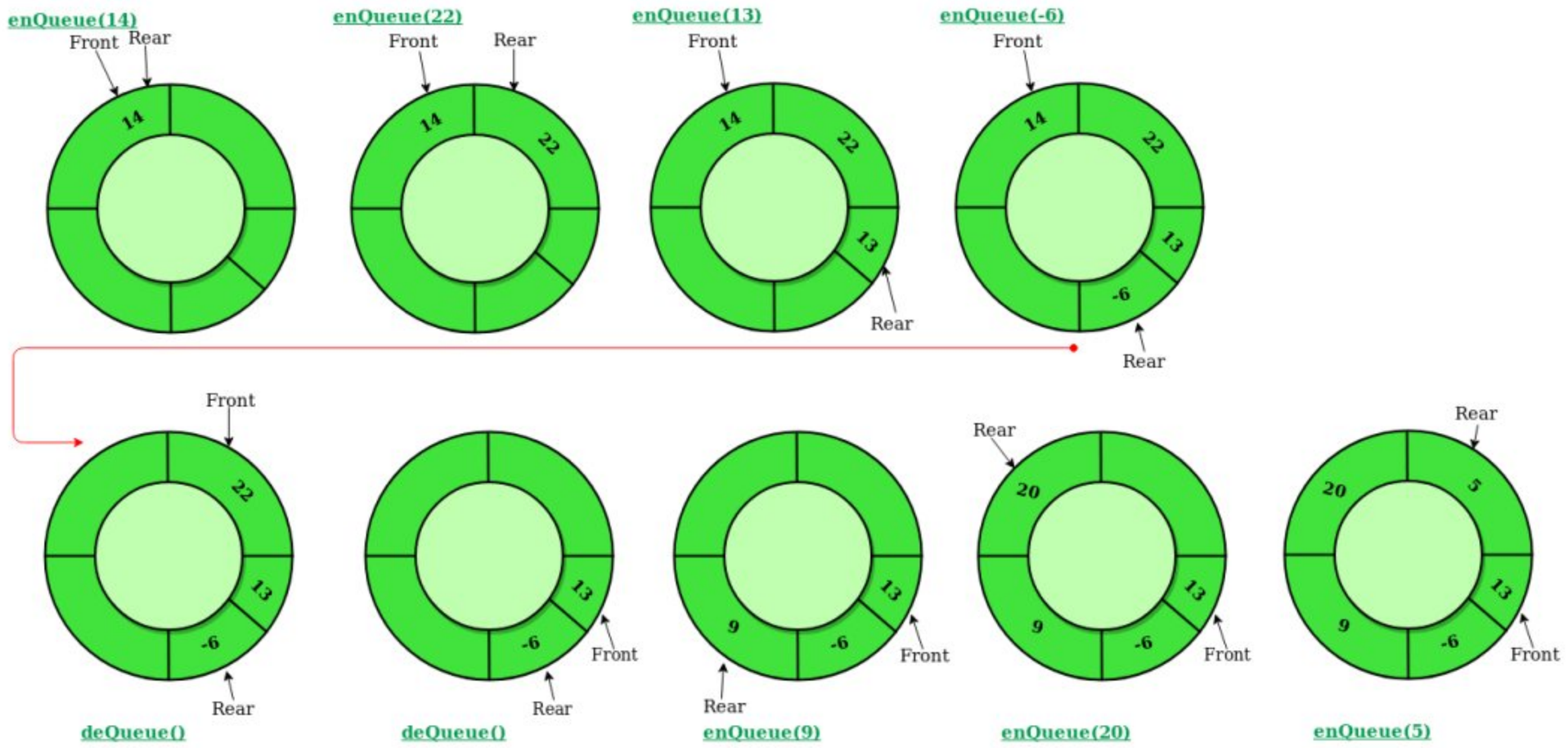
After Pass 4: {5, 6, 11, 12, 13}

Q 6: write a program to insert or delete item from circular queue.

Circular Queue is a linear data structure in which the operations are performed based on FIFO (First In First Out) principle and the last position is connected back to the first position to make a circle. It is also called **'Ring Buffer'**.



In a normal Queue, we can insert elements until queue becomes full. But once queue becomes full, we cannot insert the next element even if there is a space in front of queue.




```
//Circular Queue implementation using Array.
class CircularQueue {
int maxSize;
int rear = -1;
int front = -1;
int arr[];

public CircularQueue(int n) {

maxSize = n;
arr = new int[maxSize];

}

public boolean isEmpty() {
if (rear == -1 && front == -1) {
return true;
}
return false;

}
```

```
// Inserting element at rear end in Circularqueue.
public void enQueue(int item) {

if (((rear + 1) % maxSize) == front) {
System.out.println("Queue is overflow ");
return;
}
else if (isEmpty()) {
front = rear = 0;
}
else {
rear = (rear + 1) % maxSize;
}

arr[rear] = item;

System.out.println(item + " is inserted at position " + rear);

}

// Deleting element in CircularQueue from front end.
public void deQueue() {
int pos;
int item;
if (isEmpty()) {
System.out.println("Queue is under flow");
return;
}
else if (rear == front) {
pos = front;
item = arr[front];
rear = front = -1;
}
else {
pos = front;
item = arr[front];
front = (front + 1) % maxSize;
}
System.out.println(item + " is removed from position " + pos);

}
```