**Sec-A**

Q.1 Explain the usage of stack in recursive algorithm implementation?

In recursive algorithms, stack data structures is used to store the return address when a recursive call is encountered and also to store the values of all the parameters essential to the current state of the procedure.

Q.2 List out Applications of queue

  Operating systems often maintain a queue of processes that are ready to execute or that are waiting for a particular event to occur.

Computer systems must often provide a “h processes, two programs, or even two systems. This holding area is usually called a “buffer”       and   is   often   implemented as a queue.

Q.3 What are applications of stack?

Conversion of expression

  Evaluation of expression

  Parentheses matching

  Recursion

Q.4 Define double circularly linked list?

In a doubly linked list, if the last node or pointer of the list, point to the first element of the list, then it is a circularly linked list.

Q.5 How to implement stack using singly linked list

 Stack is an Last In First Out (LIFO) data structure. Here , elements are inserted from one end called push operation and the same elements are deleted from the same end called pop operation

 So, using singly linked list stack operations are performed in the front or other way ew can perform rear end also.

**Sec-B** (4\*2.5=10 Marks)

Q.1 Define Bubble sort. Explain with example and algorithm.

 Bubble sort is the one of the easiest sorting method. In this method each data item is compared with its neighbor and if it is an descending sorting , then the bigger number is moved to the top of all

The smaller numbers are slowly moved to the bottom position, hence it is also called as the exchange sort.

take an unsorted array for our example. Bubble sort takes Ο(n2) time so we're keeping it short and precise.

Bubble Sort

Bubble sort starts with very first two elements, comparing them to check which one is greater.

Bubble Sort

In this case, value 33 is greater than 14, so it is already in sorted locations. Next, we compare 33 with 27.

Bubble Sort

We find that 27 is smaller than 33 and these two values must be swapped.

Bubble Sort

The new array should look like this −

Bubble Sort

Next we compare 33 and 35. We find that both are in already sorted positions.

Bubble Sort

Then we move to the next two values, 35 and 10.

Bubble Sort

We know then that 10 is smaller 35. Hence they are not sorted.

Bubble Sort

We swap these values. We find that we have reached the end of the array. After one iteration, the array should look like this −

Bubble Sort

To be precise, we are now showing how an array should look like after each iteration. After the second iteration, it should look like this −

Bubble Sort

Notice that after each iteration, at least one value moves at the end.

Bubble Sort

And when there's no swap required, bubble sorts learns that an array is completely sorted.

Bubble Sort

Now we should look into some practical aspects of bubble sort.

**Algorithm**

We assume **list** is an array of **n** elements. We further assume that **swap** function swaps the values of the given array elements.

begin BubbleSort(list)

for all elements of list

if list[i] > list[i+1]

swap(list[i], list[i+1])

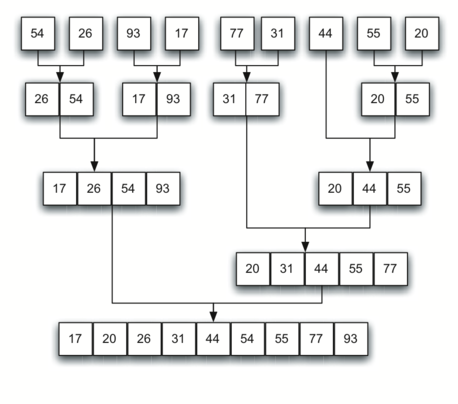
end if

end for

return list

Q.2 Define merge sort with taking any example.

Merge sort is based on divide and conquer method. It takes the list to be stored and divide it in half to create two unsorted lists.The two unsorted lists are then sorted and merge to get a sorted list



Q.3 how to implement queue using stacks.

**Method 1 (By making enQueue operation costly)** This method makes sure that oldest entered element is always at the top of stack 1, so that deQueue operation just pops from stack1. To put the element at top of stack1, stack2 is used.

**Method 2 (By making deQueue operation costly)**In this method, in en-queue operation, the new element is entered at the top of stack1. In de-queue operation, if stack2 is empty then all the elements are moved to stack2 and finally top of stack2 is returned.

Q.4 Mention the various types of searching techniques in C.

Linear search:

A simple approach is to do **linear search**, i.e

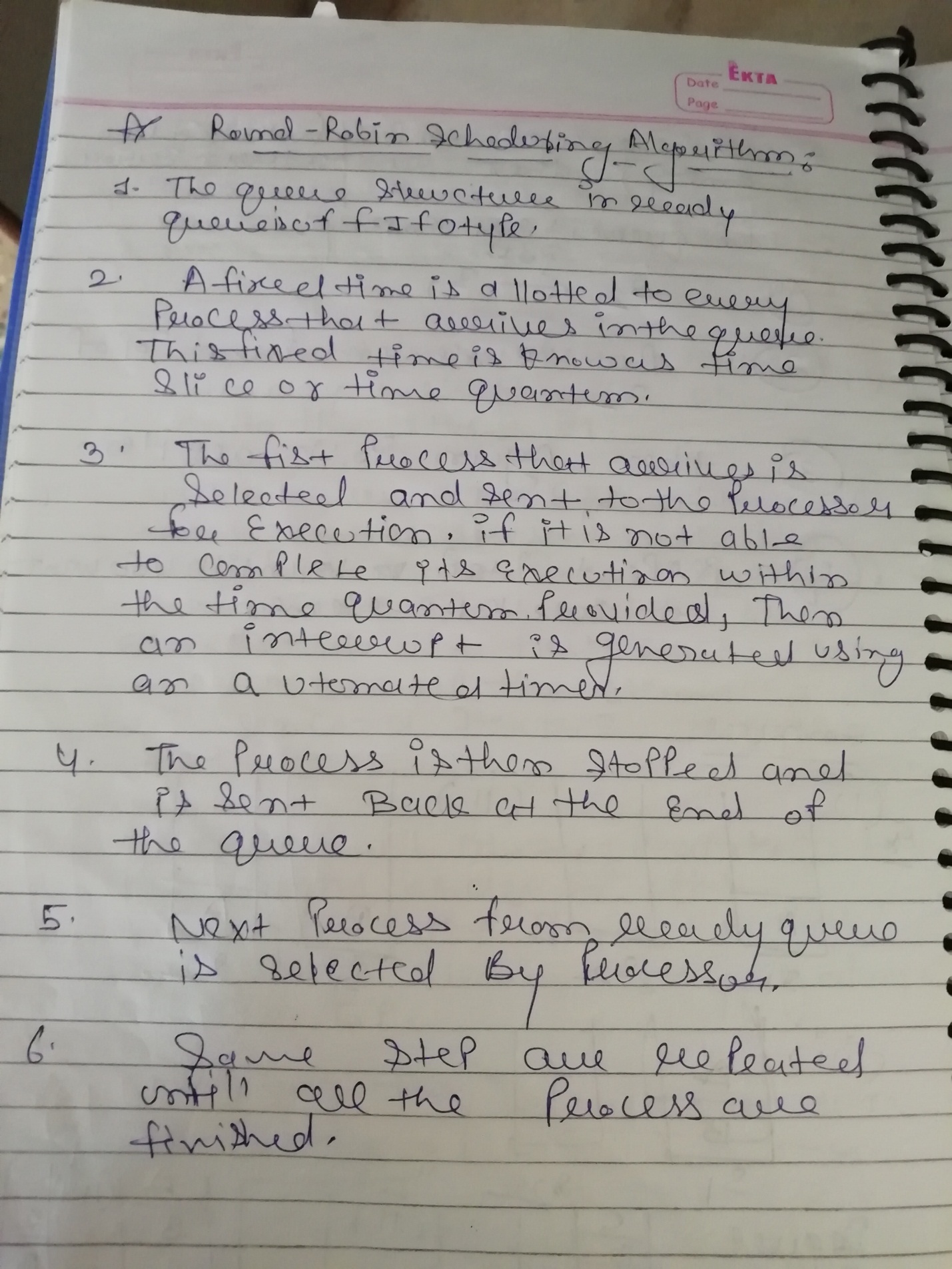
* Start from the leftmost element of arr[] and one by one compare x with each element of arr[]
* If x matches with an element, return the index.
* If x doesn’t match with any of elements, return -1.

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.

Q.5 Give a brief working of queue in round robin scheduling algorithm

Solution:



Q.6 What are the two operations of Stack? Explain any one with example and algorithm working.

Solution:

PUSH

POP

| **Concept** | **Definition** |
| --- | --- |
| Stack Push | The procedure of inserting a new element to the top of the stack is known as **Push Operation** |
| Stack Overflow | Any attempt to insert a new element in already full stack is results into Stack Overflow. |
| Stack Pop | The procedure of removing element from the top of the stack is called **Pop Operation**. |
| Stack Underflow | Any attempt to delete an element from already empty stack results into Stack Underflow. |

**Sec-C**

Q.1 Convert (a+(b\*(c-d)+(e)-(f\*g))) into postfix notation. write down the algorithm of infix to prefix conversion

Solution:

a b c d - \* e + f g \* - +

1. Step 1. Push “)” onto STACK, and add “(“ to end of the A
2. Step 2. Scan A from right to left and repeat step 3 to 6 for each element of A until the STACK is empty
3. Step 3. If an operand is encountered add it to B
4. Step 4. If a right parenthesis is encountered push it onto STACK
5. Step 5. If an operator is encountered then:
6. a. Repeatedly pop from STACK and add to B each operator (on the top of STACK) which has same
7. or higher precedence than the operator.
8. b. Add operator to STACK
9. Step 6. If left parenthesis is encontered then
10. a. Repeatedly pop from the STACK and add to B (each operator on top of stack until a left parenthesis is encounterd)
11. b. Remove the left parenthesis
12. Step 7. Exit

Q.2 What is difference between singly and doubly linked list? Explain it with example. Write down the algorithm to insert a data item in singly linked list.

The main difference between singly linked list and doubly linked list is the ability to traverse. In a single linked list, node only points towards next node, and there is no pointer to previous node, which means you can not traverse back on a singly linked list. On the other hand doubly linked list maintains two pointers, towards next and previous node, which allows you to navigate in both direction in any linked list.

**Step 1:** If AVAIL=NULL then  
Write “Availability Stack is Empty”  
           Else  
           NEW\_NODE=AVAIL  
           AVAIL = AVAIL->LINK  
**Step 2:** If FIRST = NULL then  
NEW\_NODE -> INFO = X  
           NEW\_NODE -> LINK = NULL  
           FIRST = NEW\_NODE  
           Else  
           NEW\_NODE -> INFO = X  
           NEW\_NODE -> LINK = NULL  
           SAVE = FIRST  
           Repeat while SAVE->LINK ≠ NULL  
           SAVE = SAVE->LINK  
           SAVE->LINK = NEW\_NODE  
**Step 3:** Exit

Q.3 Explain the application of stack in case of tower of Hanoi problem. Solve the tower (3, A, B, C).

N=3, Beg=A, Aux=B, End=C. also write down the algorithm.

**Step 1** − Move n-1 disks from **source** to **aux**

**Step 2** − Move nth disk from **source** to **dest**

**Step 3** − Move n-1 disks from **aux** to **dest**

A recursive algorithm for Tower of Hanoi can be driven as follows −

START

Procedure Hanoi(disk, source, dest, aux)

IF disk == 1, THEN

move disk from source to dest

ELSE

Hanoi(disk - 1, source, aux, dest) // Step 1

move disk from source to dest // Step 2

Hanoi(disk - 1, aux, dest, source) // Step 3

END IF

END Procedure

STOP