KIIT Deemed to be University

DSA Autumn Mid semester 2021 evaluation scheme

Subject Name & Code: **Data Structure & Algorithms(CS-2001)**
Applicable to Courses: B.Tech, Sem-3rd (Regular)

Full Marks=50 **Time:2 Hours**

SECTION-A(Answer All Questions. Each question carries 2 Marks)
Time:30 Minutes **(7×2=14 Marks)**

Question No	**Question Type (MCQ/SAT)**	**Question**	**CO Mapping**	**Answer Key (For MCQ Questions only)**
Q.No:1	**MCQ**	Let a two-dimensional array have a row range (45:90) and column range (50:205). The array is stored in row-major order. If the size of each element of this array is 2 bytes and base address of the array is 1000, then what will be the address of the element present at location (60, 100). A) 5750 B) 5530 C) 5630 D) 5780	C0-1	D
	MCQ	Let a two-dimensional array have a row range (45:90) and column range (50:205). The array is stored in col-major order. If the size of each element of this array is 2 bytes and base address of the array is 1000, then what will be the address of the element present at location (60, 100). A) 5750 B) 5530	C0-1	C

		C) 5630 D) 5780		
	MCQ	For one 2D matrix [15][20] each element size is ‘W’. Let in column Major Order of storing, the address of [6][8] is 4440 and the base address of matrix is at [1][1] as 4000. Find the size of each element ‘W’. A. 6 B. 4 C. 8 D. 10	C0-1	B
	MCQ	Let one n×n square matrix ARR is stored in the Column Major Order with element size as 4 bytes. If the base address is at ARR[1][1] as 1500 and the memory address of ARR[4][5] is 1608 then find out the size of matrix or value of n. a. 6 × 6 b. 12 × 12 c. 8 × 8 d. 16 × 16	C0-1	A

Q.No:2	**MCQ**	What is the time complexity of fun( ) ?
		`int fun(int n)`
		`{`
		`        int count=0;`
		`        for (int i= n; i> 0; i/=2)`
		`           for (int j=0; j<i; j++)`
		`                count+= 1;`
		`        return count;`
		`}`
		(A) O(n^2)
		(B) O(n log n)
		(C) O(n)
		(D) O(n log n log n)
			C0-2	C

	MCQ	What is the time complexity of the below function?
		`void fun(int n, int arr[])`
		`{`
		`   int i = 0, j = 0;`
		`   for (; i< n; ++i)`
		`      while (j < n && arr[i] < arr[j])`
		`         j++;`
		`}`
		(A) O(n)
		(B) O(n^2)
		(C) O(n log n)
			C0-2	A

		(D) O(n (log n) ^2)		

	MCQ	
		```
		int i, j, sum = 0;
		for (i = n / 2; i <= n; i++) {
		    for (j = 2; j <= n; j = j * 2) {
		        sum = sum + n / 2;
		    }
		  }
		  for (i = 0; i < N; i++) {
		    for (j = N; j < i; j--) {
		        sum = sum + i + j;
		    }
		}
		```
		The exact number of loop execution (in sequence of above code) and its time complexity
		a. (n/2)logn + n^2 and O(n^2)
		b. (n/2)logn-1 + n^2 and O(n^2)
		c. (n/2-1)logn + n^2 and O(nlogn)
		d. (n/2)logn + n and O(nlogn)
			C0-2	D

	MCQ	What is the time complexity of main()uj?
		```
		int fun (int n)
		{
		if (n==0)
		        return 0;
		else
		        return n+fun(n-1);
		}

		void main()
		{
		int i, n, sum;
		for (i=0; i<n; i+=2)
		        sum+=fun(i);
		}
		```
		(A) O(n^3)
		(B) O(n^2)
		(C) O(n log n)
		(D) O(n^4)
			C0-2	A

Q.No:3	**MCQ**	Assume that the operators *, -, + are left to right associative and ^ is right to left associative. The order of precedence (from highest to lowest) is ^, *, +, -. What will be the postfix expression for the given infix expression a+b*c-d^e^f	CO-4	A

		(A) abc*+def^^- (B) abc*+de^f^ (C) ab+c*d-e^f^ (D) -+a*bc^^def		
	MCQ	Assume that the STACK is initially empty. What will be the output of the following module. 1. XXX=2, YYY=5 2. PUSH (STACK, XXX) PUSH (STACK, 4) PUSH (STACK, YYY+2) PUSH (STACK, 9) PUSH (STACK, XXX+YYY) 3. WHILE TOP !=0 POP (STACK, ITEM) WRITE: ITEM END WHILE 4. Return (A) 2, 4, 7, 9, 7 (B) 7, 9, 7, 4, 2 (C) 2, 4, 9, 11 (D) 7, 9, 7, 4	CO-4	D
	MCQ	Find the postfix expression of the given infix expression. Assume that ↑ operator has the highest precedence and follows right to left associative. INFIX: (a+b) ↑ (p+q) ↑ (r*s*t) (A) ab+pq+↑rs*t*↑ (B) ab+pq+↑↑rs*t* (C) ab+pq+rs*t*↑↑ (D) ab+pq+rst**↑↑	CO-4	C
	MCQ	Assume that ↑ is the power operator and has the highest precedence. What is the output of the following postfix expression? 9 9 1 * 1 9 ↑ / + 9 - 9 + (A) 19 (B) 10 (C) 18 (D) 81	CO-4	C

Q.No:4	**MCQ**	Given the following sequence of letters and asterisks: EAS*Y*QUE***ST***IO*N*** Consider the STACK data structure, supporting two operations PUSH and POP. Suppose that for the above sequence, each letter (such as E) corresponds to a PUSH of that letter onto the STACK and each asterisk (*) corresponds a POP operation on the STACK. What will be the sequence of values returned by the POP operations. (A) SYEUQTSAOINE (B) SYEUQTSAONIE (C) SYEUQTSAOENI (D) SYEUQTSAOEIN	CO-4	B
	MCQ	Given the following sequence of letters and asterisks: EAS*Y*QUE***ST***IO*N*** Consider the QUEUE data structure, supporting two operations INSERT and DELETE. Suppose that for the above sequence, each letter (such as E) corresponds to an INSERT of that letter into the QUEUE and each asterisk (*) corresponds a DELETE operation on the queue. What will be the sequence of values returned by the DELETE operations. (A) EASYQUESTOIN (B) EASYQUESTNIO (C) EASYQUESTNOI (D) EASYQUESTION	CO-4	D

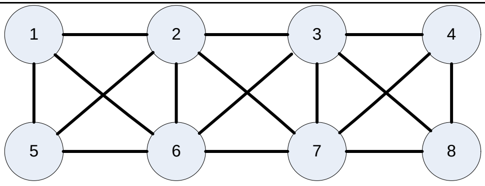
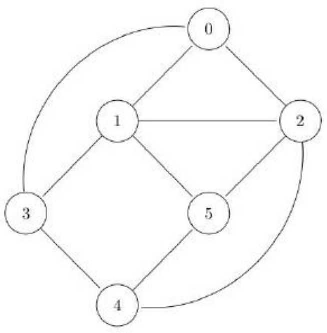
	MCQ	Let Q be a QUEUE and S be a STACK. Assume that Q and S are initially empty. What is the last integer value printed by the given code ?
		Enqueue (Q, 8);
		Enqueue (Q, 3);
		Push (S, 7);
		Push (S, 9);
		for (i=0; i<5; i++)
		    {
		    printf (“%d”, Dequeue (Q));
		    printf (“%d”, Pop (S));
		    Enqueue (Q, i);
		    Push (S, i+5);
		    }
		(A) 4
		(B) 9
			CO-4	C

		(C) 8 (D) 2		

	MCQ	Let Q be a QUEUE and S be a STACK. Assume that Q and S are initially empty. What is the last integer value printed by the given code ?
		Enqueue (Q, 8);
		Enqueue (Q, 3);
		Push (S, 7);
		Push (S, 9);
		for (i=0; i<5; i++)
		{
		printf ("%d", Pop (S));
		printf ("%d", Dequeue (Q));
		Enqueue (Q, i);
		Push (S, i+5);
		}
		(A) 4
		(B) 9
		(C) 8
		(D) 2
			CO-4	D

Q.No:5	**MCQ**	Hash table size=13, Hash function = H(x)= x mod 13, Collision resolution = quadratic probing=h+i^2 Keys=10, 100, 32, 45, 58, 126, 3, 29, 200, 400, 0, 21, 15. The key 15 will be stored at which location? A) 9 B) 10 C) 11 D) 12	CO-5	D
	MCQ	Suppose we are implementing quadratic probing with a hash function H(x)=x mod 100. If an element with key 9999 is inserted and the first three locations attempted are already occupied, then the next call that will be tried is: A) 99 B) 8 C) 4 D) 3	CO-5	B
	MCQ	Given the input {71, 23, 73, 99, 44, 79, 89} and a hash function h(x)=x mod 10 and quadratic probing, then in which location the last element will be placed? A) 2 B) 6 C) 7	CO-5	D

		D) 8		
	MCQ	Given the input {37, 38, 72, 48, 98, 11, 56} and a hash function h(x)=x mod 7 with linear probing, then in which location key 11 will be placed? (The table size is 7 indexed from 0 to 6.) A) 3 B) 4 C) 5 D) 6	CO-5	C
Q.No:6	**MCQ**	Which character will be placed in the root node of the expression tree for the following expression.? a/b-c/((d+e)-f)ˆg*h/p+k A) + B) - C) / D) ^	CO-4	A
	MCQ	Which character will be placed in the root node of the expression tree for the following expression.? a/b+c/((d+e)-f)ˆg*h/p-k A) + B) - C) / D) ^	CO-4	B
	MCQ	What is the post-order traversal of a binary tree whose in-order and pre-order traversals are as given below? INORDER: F, C, E, D, B, A, I, K, G, J, H PREORDER: A, B, C, F, D, E, G, I, K, H, J A) POSTORDER: F, E, D, C, B, K, I, J, H, G, A B) POSTORDER: F, C, E, D, B, K, J, I, H, G, A C) POSTORDER: F, C, E, D, B, K, G, I, H, J, A D) POSTORDER: A, C, E, D, B, K, J, I, H, G, F	CO-4	A
	MCQ	What is the pre-order traversal of a binary tree whose in-order and post-order traversals are as given below? INORDER: F, C, E, D, B, A, I, K, G, J, H POSTORDER: F, E, D, C, B, K, I, J, H, G, A A) PREORDER: A, B, C, F, D, E, G, I, K, J, H B) PREORDER: A, B, C, F, D, E, G, I, K, H, J C) PREORDER: A, B, C, F, D, E, G, K, I, H, J D) PREORDER: A, B, C, F, D, G, E, I, K, H, J	CO-4	B
Q.No:7	**MCQ**	Let G=(V, E) is an undirected graph where each edge has a unique cost. Consider the following statements and choose which one of the following is TRUE.	CO-4	C

		I. For a given pair of vertices Vi and Vj, there always exist a unique shortest path between them. II. If the weights of the graph are multiplied by a positive constant, the shortest paths remain unchanged. A) I and II B) Only I C) Only II D) None		
	MCQ	 Identify valid BFS search sequences A) 1, 5, 2, 3, 4, 7, 6, 8 B) 1, 2, 5, 6, 7, 4, 3, 8 C) 1, 2, 5, 6, 7, 3, 4, 8 D) 1, 5, 2, 7, 6, 3, 4, 8	CO-4	C
	MCQ	Which of the following is an advantage of adjacency list representation over the adjacency matrix representation of a graph? A. In adjacency list representation, space is saved for sparse graphs. B. DFS and BSF can be done in O(V + E) time for adjacency list representation. These operations take O(V^2) time in adjacency matrix representation. Here V and E are the number of vertices and edges respectively. C. Adding a vertex in adjacency list representation is easier than adjacency matrix representation. D. All of the above	CO-4	D
	MCQ	Identify valid DFS search sequences. The search starts at vertex 0 and lexicographic ordering is assumed for the edges originating from each vertex. 	CO-4	A

		A) 0 1 2 4 3 5 B) 0 1 2 5 4 3 C) 0 1 2 3 4 5 D) 0 1 3 4 2 5		

SECTION-B(Answer Any Three Questions. Each Question carries 12 Marks)

Time: 1 Hour and 30 Minutes
(3×12=36 Marks)

Question No	**Question**	**CO Mapping (Each question should be from the same CO(s))**
Q.No: 8	**Evaluation scheme: Correct answer will be given full marks and partial marks to be awarded depending on the correctness of the steps.** Write the code to implement a queue using two stacks. [5] Sol: **a. code to implement a queue using two stacks** *enQueue(q, x):* • *While stack1 is not empty, push everything from stack1 to stack2.* • *Push x to stack1 (assuming size of stacks is unlimited).* • *Push everything back to stack1.* *Here time complexity will be O(n)* *deQueue(q):* • *If stack1 is empty then error* • *Pop an item from stack1 and return it* *Here time complexity will be O(1)* // Program to implement Queue using Stack in C. #include<stdio.h> #define N 5	CO-4

```
int stack1[5], stack2[5]; // declaration of two stacks
// declaration of top variables.
int top1=-1, top2=-1;
int count=0;
// inserting the elements in stack1.
void push1(int data)
{
// Condition to check whether the stack1 is full or not.
 if(top1==N-1)
{
   printf("\n Stack is overflow...");
}
else
{
   top1++;   // Incrementing the value of top1
   stack1[top1]=data;  // pushing the data into stack1
}
}
// Removing the elements from the stack1.
int pop1()
{
// Condition to check whether the stack1 is empty or not.
if(top1==-1)
{
   printf("\nStack is empty..");
}
else
{
   int a=stack1[top1];  // Assigning the topmost value of stack1 to 'a' variable.
   top1--;  // decrementing the value of top1.
   return a;
}
}
// pushing the data into the stack2.
void push2(int x)
{
//  Condition to check whether the stack2 is full or not
if(top2==N-1)
{
   printf("\nStack is full..");
}
else
{
    top2++;   // incrementing the value of top2.
    stack2[top2]=x;  // assigning the 'x' value to the Stack2

}
```

```
}
// Removing the elements from the Stack2
int pop2()
{
   int element = stack2[top2];  // assigning the topmost value to element
   top2--;  // decrement the value of top2
   return element;
}
void enqueue(int x)
{
    push1(x);
    count++;
}
void dequeue()
{
   if((top1==-1) && (top2==-1))
{
   printf("\nQueue is empty");
}
else
{
  for(int i=0;i<count;i++)
  {
     int element = pop1();
     push2(element);
  }
int b= pop2();
printf("\nThe dequeued element is %d", b);
printf("\n");
count--;
for(int i=0;i<count;i++)
{
   int a = pop2();
   push1(a);
}
}}
void display()
{
   for(int i=0;i<=top1;i++)
  {
     printf("%d , ", stack1[i]);
  }
}
void main()
{
   enqueue(10);
   enqueue(20);
```

```
        enqueue(30);
        dequeue();
        enqueue(40);
        display();
}
```

Apply the following sequence of operations stepwise in a normal queue as well as queue using two stacks. And analyze the performance in terms of their time complexity. [4]

enqueue 1, 2, 3

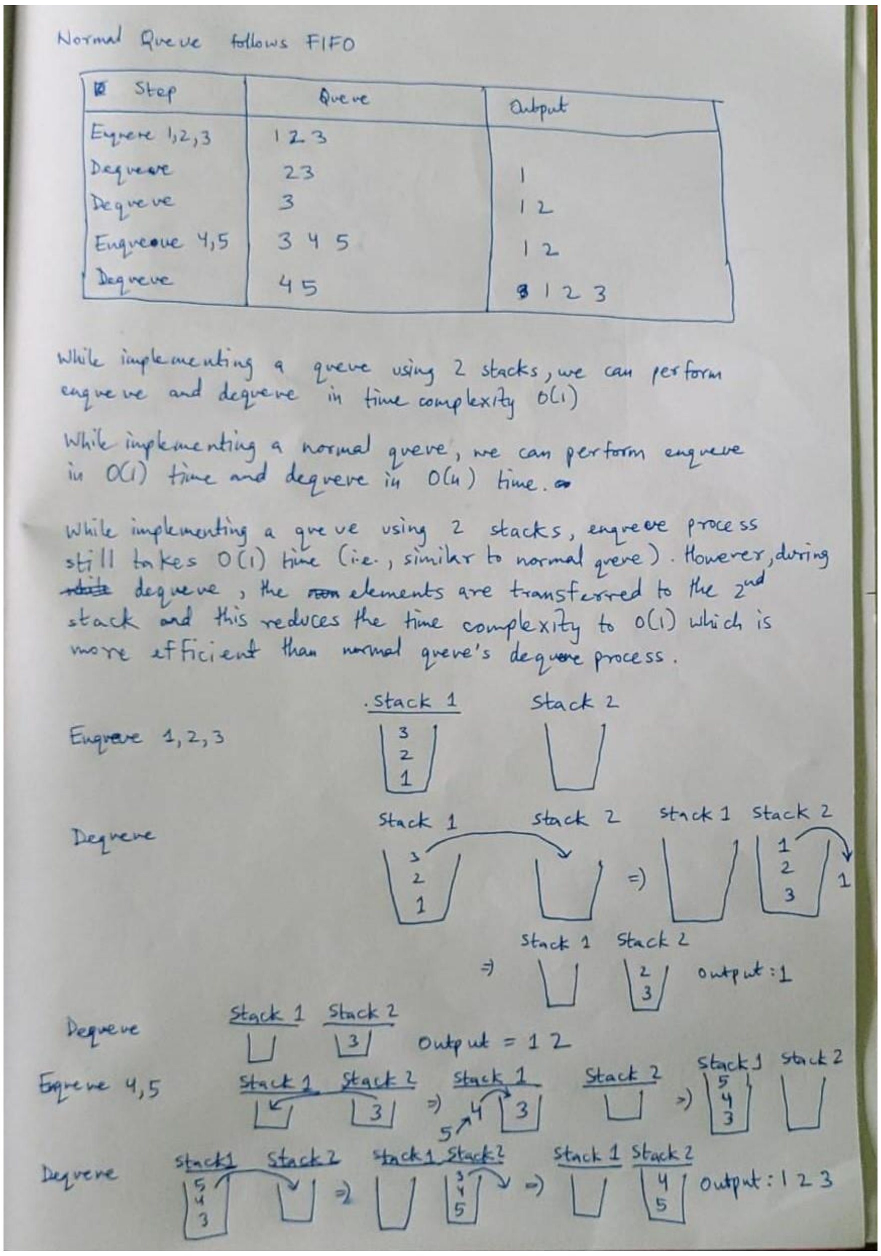
dequeue

dequeue

enqueue 4, 5

dequeue

Sol:



Normal Queue follows FIFO

Step	Queue	Output
Enqueue 1,2,3	1 2 3	
Dequeue	2 3	1
Dequeue	3	1 2
Enqueue 4,5	3 4 5	1 2
Dequeue	4 5	~~3~~ 1 2 3

While implementing a queue using 2 stacks, we can perform enqueue and dequeue in time complexity O(1)

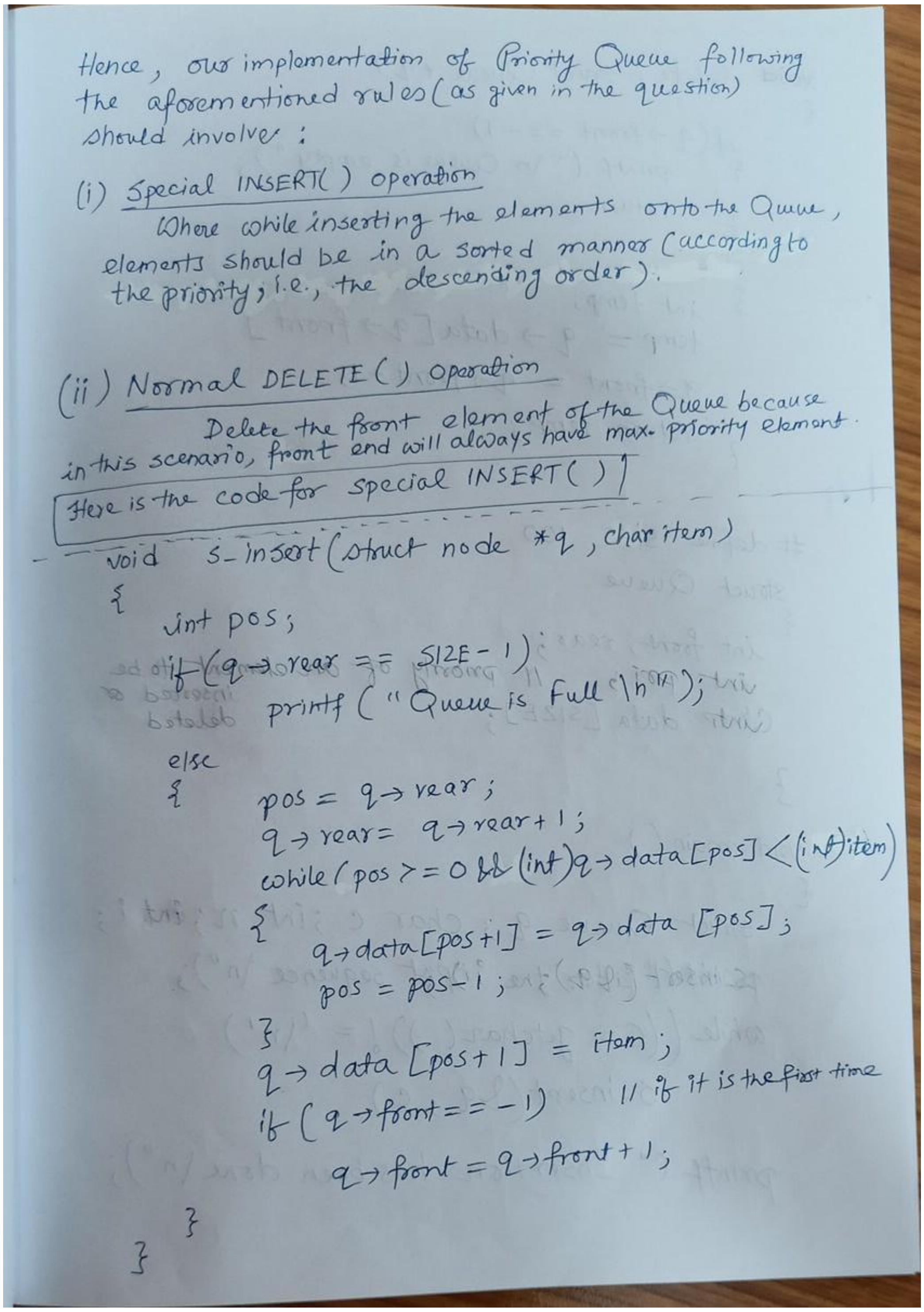
While implementing a normal queue, we can perform enqueue in O(1) time and dequeue in O(n) time.

While implementing a queue using 2 stacks, enqueue process still takes O(1) time (i.e., similar to normal queue). However, during ~~[illegible]~~ dequeue, the ~~[illegible]~~ elements are transferred to the 2nd stack and this reduces the time complexity to O(1) which is more efficient than normal queue's dequeue process.

Write the code to implement a priority queue where INSERT() will insert a letter with its priority value at the appropriate position and DELETE() will delete the letter with maximum priority inserted so far.

Note: Consider the priority value of A to Z as 1 to 26. A higher number is having higher priority. [3]

Sol:



Hence, our implementation of Priority Queue following the aforementioned rules (as given in the question) should involve :

(i) Special INSERT( ) operation

Where while inserting the elements onto the Queue, elements should be in a sorted manner (according to the priority; i.e., the descending order).

(ii) Normal DELETE ( ) operation

Delete the front element of the Queue because in this scenario, front end will always have max. priority element.

Here is the code for special INSERT ( )

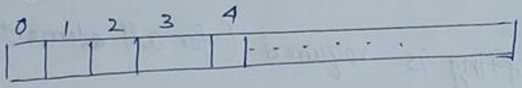
```
void s_insert(struct node *q , char item)
{
    int pos;
    if(q->rear == SIZE-1)
        printf ("Queue is Full \n");
    else
    {
        pos = q->rear;
        q->rear= q->rear+1;
        while(pos >= 0 && (int)q->data[pos] < (int)item)
        {
            q->data[pos+1] = q->data [pos];
            pos = pos-1;
        }
        q->data [pos+1] = item;
        if (q->front==-1)        // if it is the first time
            q->front = q->front+1;
    }
}
```

```
    printf ("Enter the no. of elements to be deleted \n");
    scanf ("%d", &n);
    for (i=1; i<=n; i++)
    {   d= delete(&q);
        printf ("Item got deleted \t %d\n", d );

    }

}
```

Ex. We have the following i/p sequence

C A F I H G B D E J K

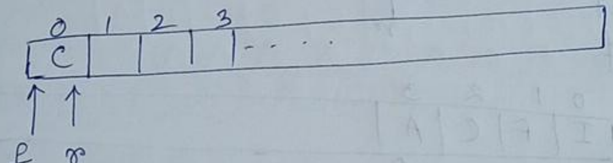
✓ Initially the array is empty, i.e.,



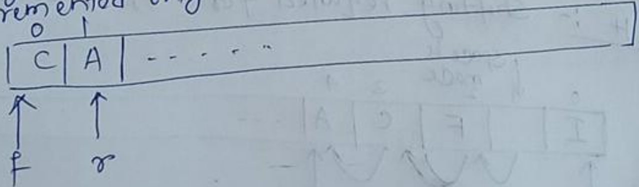
front = rear = -1

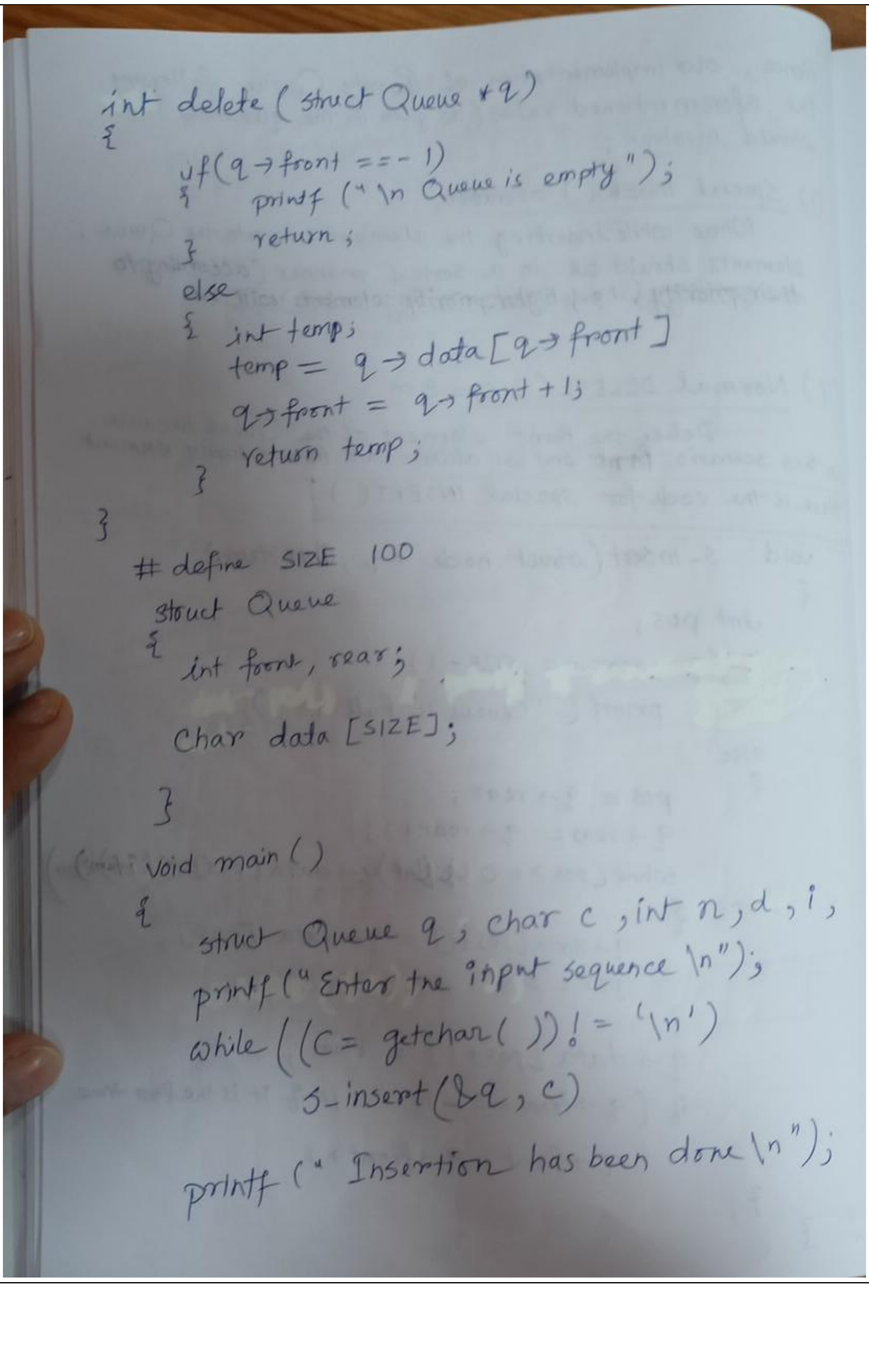
✓ The i/p C is read and going to be inserted.

✓ front and rear both are incremented

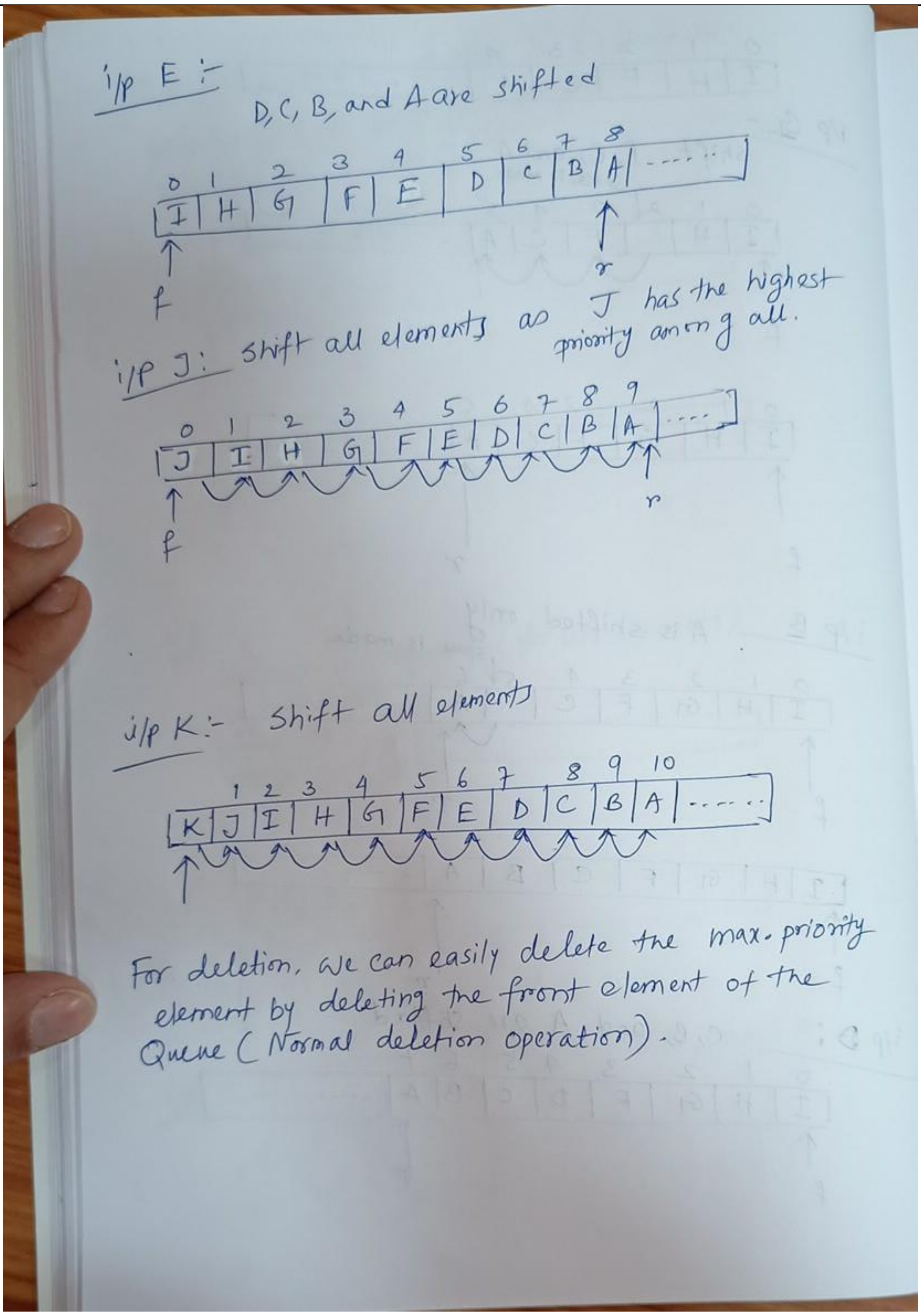


Then i/p A is read and inserted.
No shifting is required as priority of C is > priority of A
rear is incremented only





```
int delete ( struct Queue *q)
{
        if(q->front ==-1)
        {    printf ("\n Queue is empty");
             return ;
        }
        else
        {  int temp;
           temp = q->data[q->front]
           q->front = q->front + 1;
           return temp;
        }
}
   # define SIZE 100
    struct Queue
    {
       int front, rear;

       Char data [SIZE];

    }
  void main ( )
  {
        struct Queue q, char c ,int n, d, i,
        printf (" Enter the input sequence \n");
        while ((C= getchar ( ))!= '\n')
              S_insert(&q, c)

        printf (" Insertion has been done \n");
```



i/p E :-

D, C, B, and A are shifted

0	1	2	3	4	5	6	7	8	
I	H	G	F	E	D	C	B	A	.....
f								r	

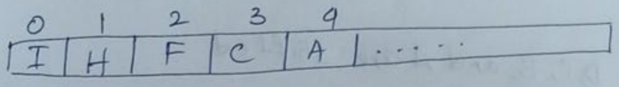
i/p J: shift all elements as J has the highest priority among all.

0	1	2	3	4	5	6	7	8	9	
J	I	H	G	F	E	D	C	B	A	....
f									r	

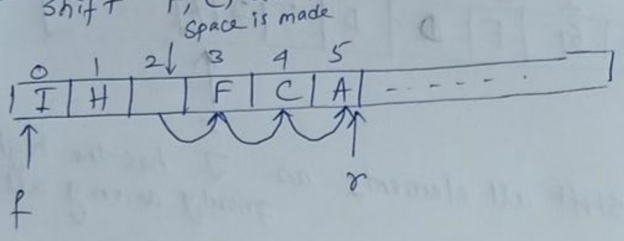
i/p K:- Shift all elements

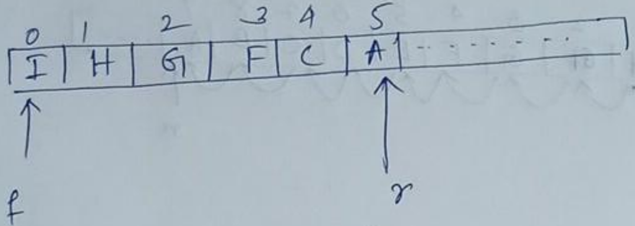
	1	2	3	4	5	6	7	8	9	10	
K	J	I	H	G	F	E	D	C	B	A	......

For deletion, we can easily delete the max. priority element by deleting the front element of the Queue (Normal deletion operation).

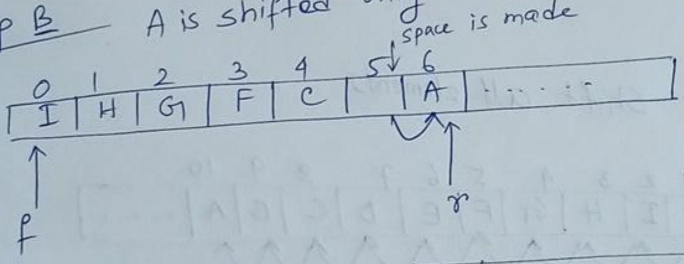


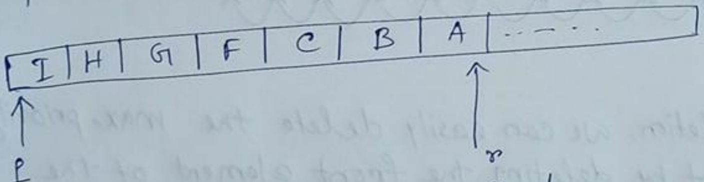
i/p G:- shift F, C, and A



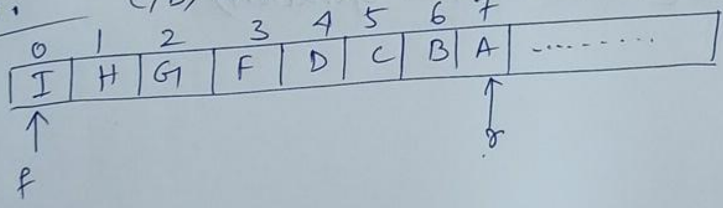


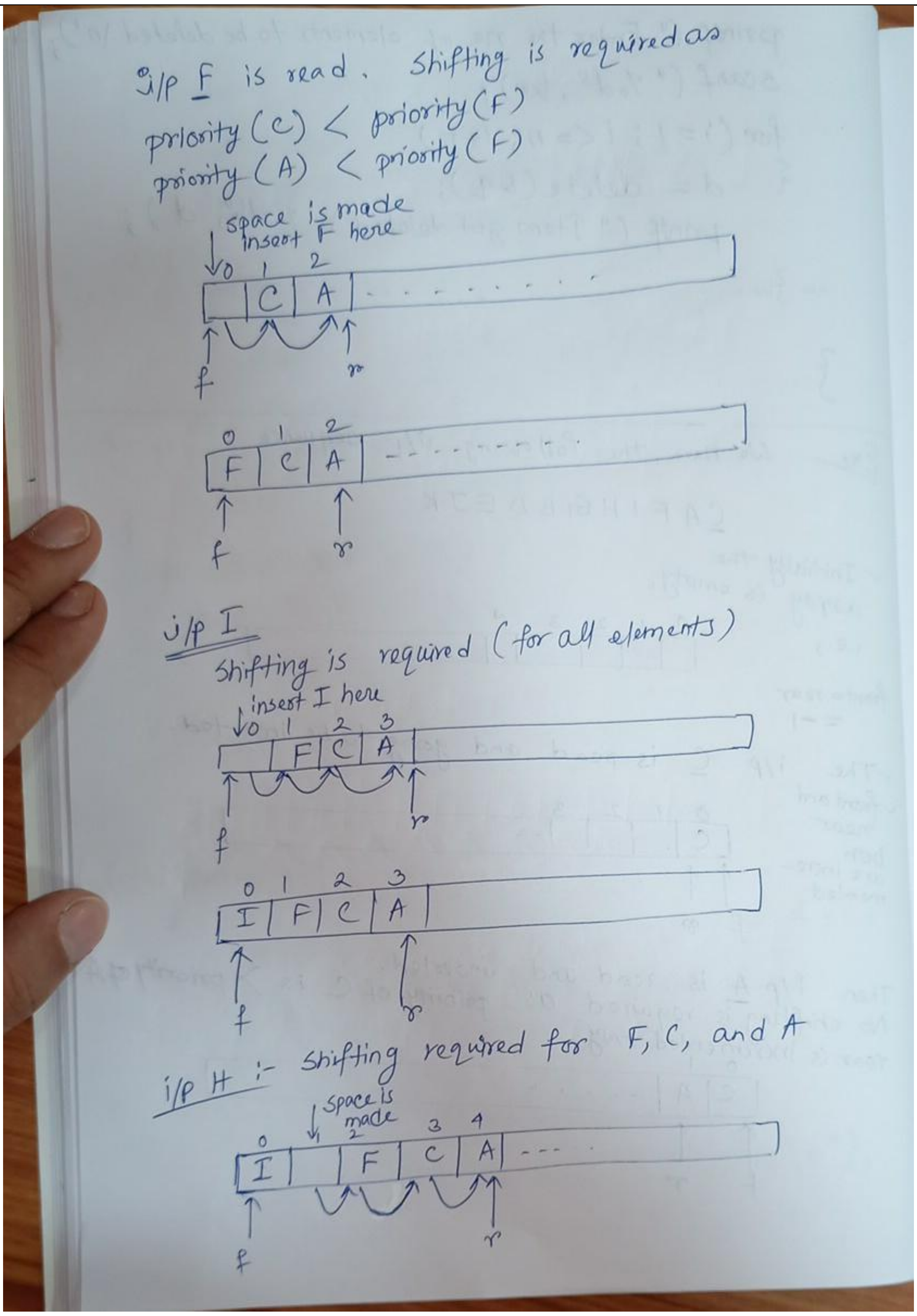
i/p B A is shifted only





i/p D: C, B, and A are shifted





i/p F is read. Shifting is required as
priority (C) < priority (F)
priority (A) < priority (F)

space is made
insert F here
0
1
2
C
A
f
r
0
1
2
F
C
A
f
r

i/p I
Shifting is required (for all elements)

insert I here
0
1
2
3
F
C
A
f
r
0
1
2
3
I
F
C
A
f
r

i/p H :- Shifting required for F, C, and A

space is made
0
1
2
3
4
I
F
C
A
f
r

Evaluation scheme: Correct answer will be given full marks and partial marks to be awarded depending on the correctness of the steps.

Write the code to implement a queue using a circular singly linked list. Show how you can perform enqueue and dequeue operations in O(1) time. [4]

Sol:

Both EnQueue and DeQueue operation cannot be performed in O(1) time in case of Single Circular List Representation.

	Either of the operations can be performed.
	`# include<stdio.h>`
	`# include<stdlib.h>`
	`struct CList{`
	`int Data;`
	`struct CList *next;`
	`};`
	
	`struct Q{`
	`struct CList *CL;`
	`struct CList *rear;`
	`};`
	
	`struct CList* init_CList(int data)`
	`{`
	`struct CList *temp = (struct CList*) malloc (sizeof(struct CList));`
	`temp->Data = data;`
	`temp->next = temp;`
	`return temp;`
	`};`
	
	`struct Q* init_Q(int data)`
	`{`
	`struct Q *Q_temp = (struct Q*) malloc (sizeof(struct Q));`
	`Q_temp->CL = init_CList(data);`
	`Q_temp->rear = Q_temp->CL;`
	`return Q_temp;`
	
	`}`
	
	`void print(struct CList *C)`
	`{`
	`struct CList *temp = C;`
	`do`
	`{`
	`printf("->%d", temp->Data);`
	`temp = temp->next;`
	`} while(temp != C);`
	`printf("--->\n");`
	`}`
	
	`void printQ(struct Q *q)`
	`{`
	`print(q->CL);`
	`}`
	
	`struct CList* insertR(struct CList *C,int data)`
	`{`	

```
struct CList *temp = init_CList(data);
temp->next = C;
C->next = temp;
C = temp;
return C;

}
struct Q* EnQ(struct Q *q ,int data)
{
struct CList* tempNode = (struct CList*) malloc (sizeof(struct CList));
tempNode->Data = data;
q->rear->next = tempNode;
tempNode->next = q->CL;
q->rear = tempNode;


return q;

}

void DeQ(struct Q *q)
{
struct CList *temp = q->CL;
int data = temp->Data;
q->rear->next = q->CL->next;
q->CL = q->CL->next;
free(temp);
}


int main()
{
struct Q *q = init_Q(10);
q = EnQ(q,20);
q = EnQ(q,30);
q = EnQ(q, 40);
printQ(q);
DeQ(q);
printQ(q);
DeQ(q);
printQ(q);
DeQ(q);
printQ(q);
q = EnQ(q, -40);
printQ(q);

return 0;
}
```

Write the code to implement a stack using two queues. [5]

Sol:

```
#include <stdio.h>
#include <stdlib.h>

struct Queue{
    int a[10];
    int front, rear;
};

//Queue operations
void insert(struct Queue *q, int no);

void display(struct Queue *q);

void delete(struct Queue *q, int *no);

//Stack push operation
struct Queue push(struct Queue *q, int no);
//Stack pop operation is handled by delete operation itself.
int main(){
    struct Queue q1;
    q1.front = q1.rear = -1;
    int choice = 1, op;
    while(choice){
        printf("\nEnter 1 to enter data in Stack.\nEnter 2 to Delete element from
Stack.\nEnter 3 to Display the Stack.\nEnter 0 to Exit\nEnter your choice:\t");
        scanf("%d", &op);
        printf("\n");
        switch(op){
            case 1:{
                int no;
                printf("Enter number to be added to Stack:\t");
                scanf("%d", &no);
                q1 = push(&q1, no);
            }
            break;
            case 2:{
                int delno;
                printf("Deleting Element from Stack.\n");
                delete(&q1, &delno);
                printf("Deleted number is:\t%d\n\n", delno);
            }
            break;
            case 3:{
                display(&q1);
            }
            break;
            case 0:{
                return 0;
            }
            break;
            default:{
                printf("Please Enter a valid choice for Operation.\n\n");
            }
        }
    }
}

struct Queue push(struct Queue *q1, int no){
    struct Queue q2;
    q2.front = q2.rear = -1;
    insert(&q2, no);
    int delelem;
    while(q1->rear >= q1->front && q1->front != -1){
        delete(q1, &delelem);
        insert(&q2, delelem);
    }
    return q2;
}

void insert(struct Queue *q, int no){
    if(q->front == -1 && q->rear == -1){
        q->front = 0;
        q->rear = 0;
        q->a[q->front] = no;
    }
    else if((q->rear == 9)){
        printf("Overflow\n\n");
        return;
    }
    else{
        q->rear++;
        q->a[q->rear] = no;
    }
}

void display(struct Queue *q){
    int temp;
    if(q->front != -1){
        for(temp = q->front; temp<=q->rear; temp++){
            printf("%d ", q->a[temp]);
        }
    }
    else{
        printf("Empty stack.");
    }
    printf("\n\n");
}

void delete(struct Queue *q, int *no){
    if(q->front == -1 && q->rear == -1){
        printf("Underflow\n\n");
    }
    else if( (q->front == q->rear) || (q->front == 9) ){
        *no = q->a[q->front];
        q->front = q->rear = -1;
    }
    else{
        *no = q->a[q->front];
        (q->front)++;
    }
}
```

Apply the following sequence of operations in stack using two queues and stepwise show the output. [3]

push 1, 2, 3

pop

pop

push 4, 5

pop

Sol:

Push 1,2,3->3,2,1
Pop->2,1
Pop->1
Push 4,5->1,4,5->5,4,1
Pop->4,1

Evaluation scheme: Correct answer will be given full marks and partial marks to be awarded depending on the correctness of the steps.

Convert the given INFIX expression into POSTFIX expression using STACK.
5 *(((9 ^6* 8+3) ∗ (4 ∗ 6)) + 7). [4]

Sol:

5 *(((9 ^6* 8+3) ∗ (4 ∗ 6)) + 7))		
Symbols	**Stack**	**Postfix expression**
	(	
5	(	5
*	(*	5
(	(*(	5
(	(*((	5
(	(*(((	5
9	(*(((	59
^	(*(((^	59
6	(*(((^	596
*	(*(((*	596^

8	(*(((*	596^8
+	(*(((+	596^8*
3	(*(((+	596^8*3
)	(*((	596^8*3+
*	(*((*	596^8*3+
(	(*((*(	596^8*3+
4	(*((*(	596^8*3+4
*	(*((*(*	596^8*3+4
6	(*((*(*	596^8*3+46
)	(*((*	596^8*3+46*
)	(*(	596^8*3+46**
+	(*(+	596^8*3+46**
7	(*(+	596^8*3+46**7
)	(*	596^8*3+46**7+
)	Empty	596^8*3+46**7+*

Evaluate the above POSTFIX expression using STACK. [4]

Sol:

596^8*3+46**7+*		
Symbol	**Stack**	**Operation**
5	**5**	
9	5, 9	
6	5, 9, 6	
^	5, 531441	9 ^ 6=531441
8	5, 531441, 8	
*	5, 4251528	531441 * 8= 4251528
3	5, 4251528, 3	
+	5, 4251531	4251528 + 3 =4251532
4	5, 4251531, 4	
6	5, 4251531, 4, 6	
*	5, 4251531,24	4 * 6 = 24
*	5, 102436744	4251532* 24=102436744
7	5, 102436744, 7	
+	5, 102436751	102436744 +7=102436751
*	512183755	102436751* 5= 512183755

Write the pseudo-code for evaluation of POSTFIX expression. [4]

Sol:

Postfix evaluation pseudo code:

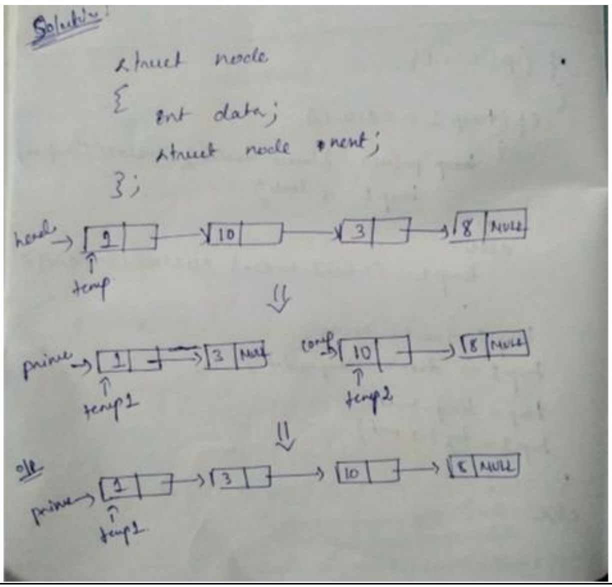
postfix: post expression

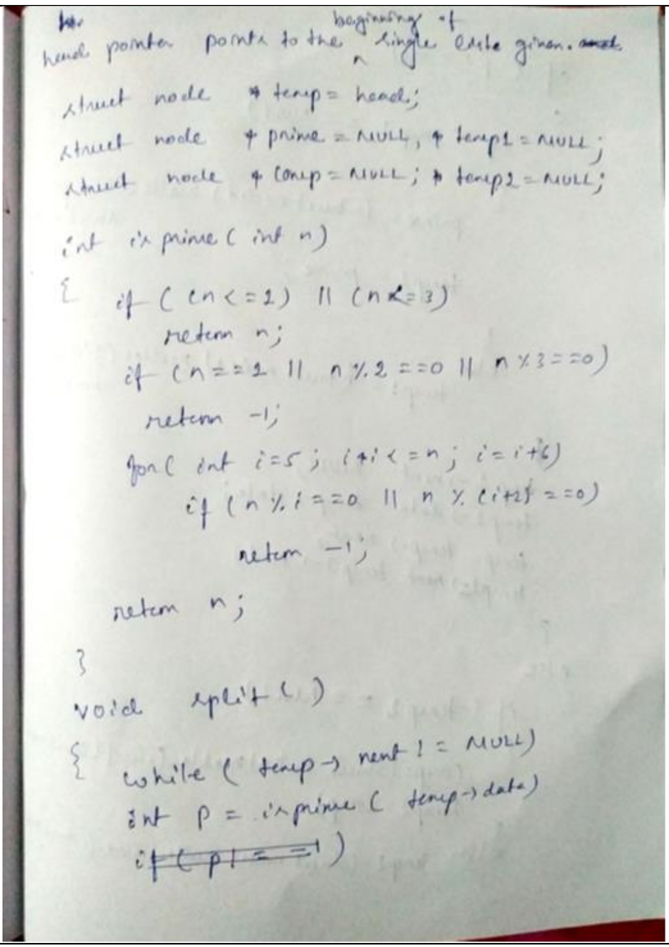
stack: initially empty stack

token: stores scanned character at each iteration

```
FOR i = 1 TO postfix.LENGTH DO
```

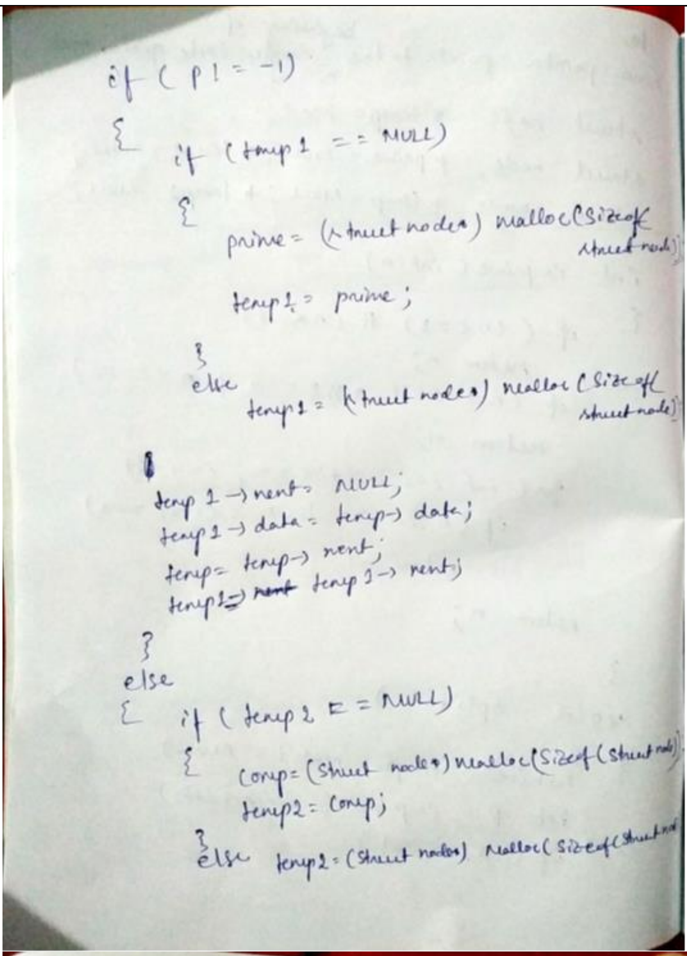
	token := postfix[i]
	IF token is operand THEN
	   PUSH token to stack
	ELSE IF token is operator THEN
	   POP operands from stack.
	   Perform the operation
	   PUSH the result to stack
	END
	END FOR	

<u>Q.No: 9</u>	**Evaluation scheme: Correct answer will be given full marks and partial marks to be awarded depending on the correctness of the steps.**
	
	Given a single list, write the code to split the list into two sublists; one with prime numbers and the other with composite numbers. Then append the composite sub-list after the prime sub-list. [4]
	Sol:
	
	Solution:
	struct node
	{
	   int data;
	   struct node *next;
	};
	 head
	1
	10
	3
	8
	NULL
	temp
	prime
	1
	3
	NULL
	comp
	10
	8
	NULL
	temp1
	temp2
	o/p
	prime
	1
	3
	10
	8
	NULL
	temp1
		CO 3, CO-4, CO 6

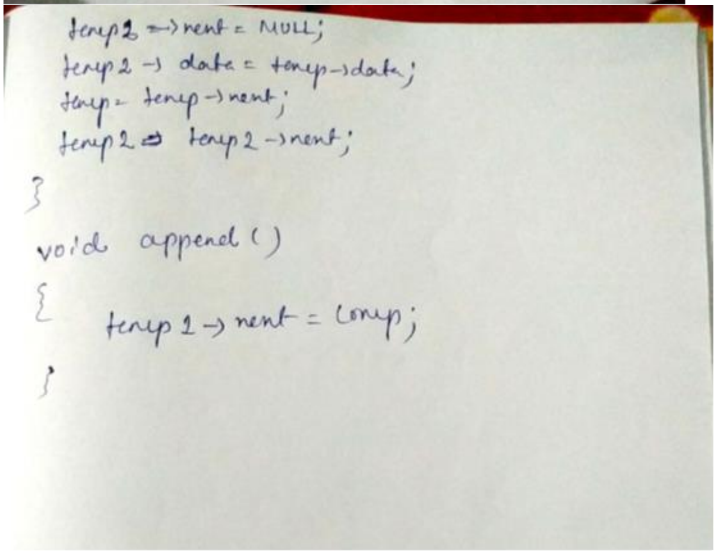
				

~~the~~ head pointer points to the beginning of single list given. ~~and~~

```
struct node  * temp= head;
struct node  * prime = NULL, * temp1 = NULL;
struct  node  * comp = NULL; * temp2 = NULL;

int  is prime ( int n)
{   if ( (n <= 1) || (n <= 3)
        return n;
    if (n == 1 || n % 2 == 0 || n % 3 == 0)
     return -1;
    for( int i=5 ; i*i <= n ; i = i+6)
        if (n % i == 0  || n % (i+2) == 0)
             return -1;
  return n;
}
void  split ( )
{  while ( temp -> next != NULL)
   int  p = isprime ( temp->data)
   if ~~( p ! = = 1~~ )
```



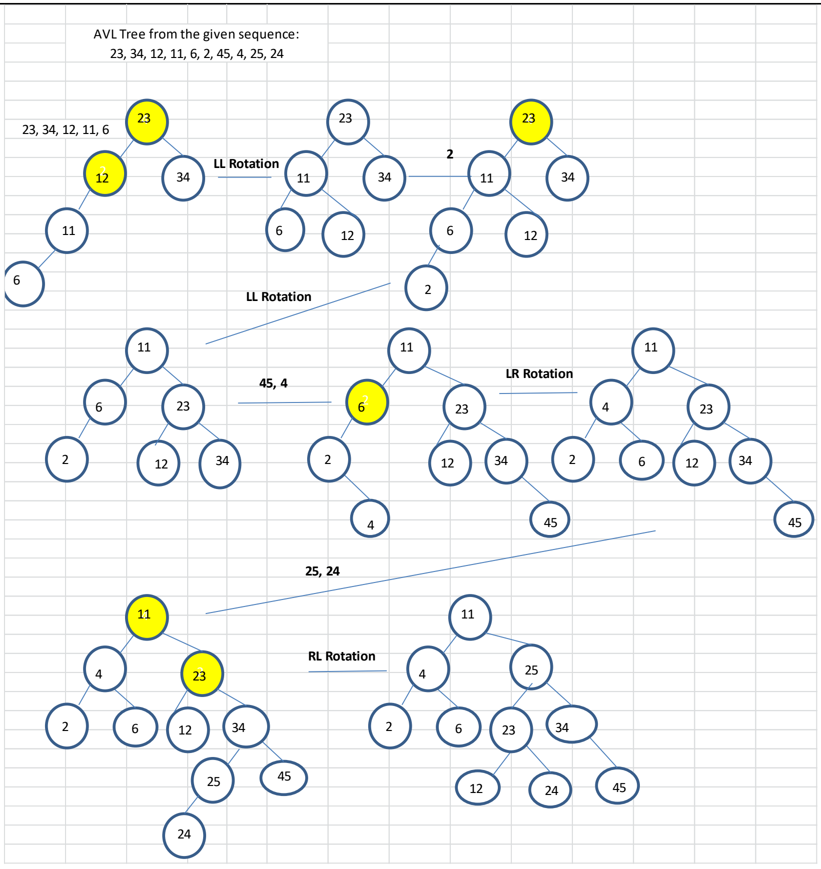
```
if ( p != -1)
{
    if (temp1 == NULL)
    {
        prime = (struct node*) malloc(sizeof(
                                   struct node));
        temp1 = prime ;
    }
    else
        temp1 = (struct node*) malloc(sizeof(
                                   struct node));

    temp1 -> next = NULL;
    temp1 -> data = temp-> data;
    temp = temp-> next;
    temp1 = ~~next~~ temp1-> next;
}
else
{   if ( temp2 == NULL)
    {
        comp = (struct node*) malloc(sizeof(struct node));
        temp2 = comp;
    }
    else temp2 = (struct node*) malloc( sizeof(struct node
```



```
    temp2 -> next = NULL;
    temp2 -> data = temp->data;
    temp = temp->next;
    temp2 = temp2 ->next;
}
void append ()
{
    temp1 -> next = comp;
}
```

Construct an AVL tree by inserting the following elements in the order of their occurrence. Clearly mention which rotations you are applying in each step. 23, 34, 12, 11, 6, 2, 45, 4, 25, 24. [5]

Sol:



Write the INORDER, PREORDER, and POSTORDER traversal of the above AVL tree. [3]

Sol:
In-order:2, 4, 6, 11, 12, 23, 24, 25, 34, 45
Pre-order:11, 4, 2, 6, 25, 23, 12, 24, 34, 45
Post-order: 2, 6, 4, 12, 24, 23, 45, 34, 25, 11

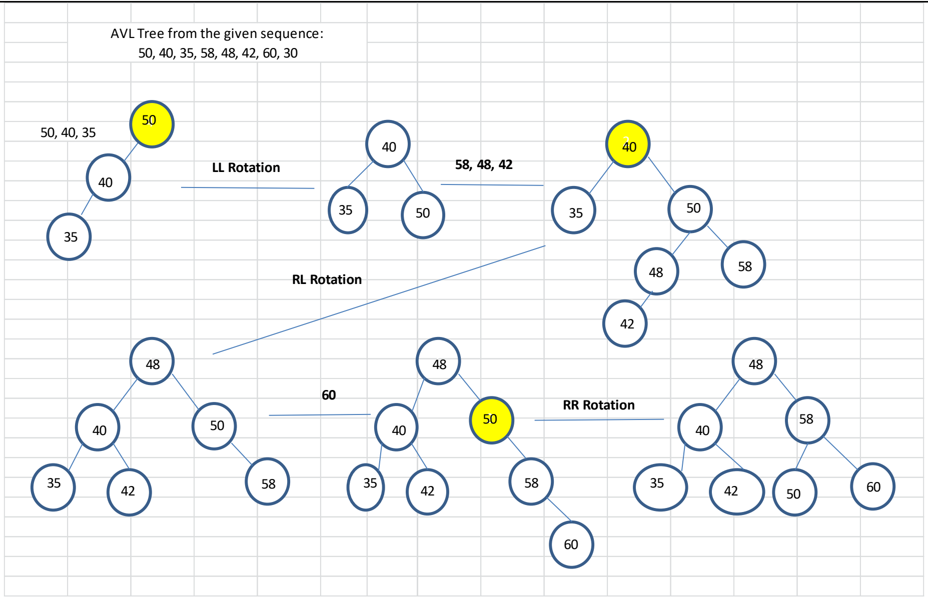
Evaluation scheme: Correct answer will be given full marks and partial marks to be awarded depending on the correctness of the steps.

Given a single linked list, write the code to split it into two sublists; one for the 1st half, and other for the 2nd half. If the number of elements are odd, the extra element should go to the 1st sublist. After splitting, append the reversed 1st list after the 2nd list.
Note: If the list is {2, 3, 5, 7, 11} then it should yield the two sublists {2, 3, 5} and {7, 11} and after append the output will be {7, 11, 5, 3, 2}.
[4]

Sol:
```
//Split the list into two parts as per the question
struct node * split( struct node *start)
{
  struct node *p, *q;
  p = q= start;
  while( q!= NULL && q->next != NULL)
  {
        q=q->next->next;
        P=p->next;
    }
    struct node *new = p->next;
    P->next = NULL;
    Return(new);
}

// Revering the first split
reverse(struct node **start)
{
 struct node *p1, *p2, *p3;
 if (start->next == NULL)
     return;
 p1 = start;
  p2 = p1->next;
  p3 = p2 -> next;

  p1->next = NULL;
  p2->next = p1;

  while(p3 != NULL)
  {
    p1 = p2;
    p2 = p3;
    p3 = p3 - >next;
    p2->next = p1;
  }
  start = p2;
}

main()
{
       struct node *start, *start_new, *p;
       //create the linked list with starting node as start

       start_new=split(&start);
       reverse(&start);
       for(p=start_new; p->next!=NULL; p=p->next);
       p->next=start;
       display(start_new); //will display the list headed by start_new
}
```

Construct an AVL tree by inserting the following elements in the order of their occurrence. Clearly mention which rotations you are applying in each step. 50, 40, 35, 58, 48, 42, 60, 30. [5]

Sol:



Write the INORDER, PREORDER, and POSTORDER traversal of the above AVL tree. [3]

Sol:

In-order:35, 40, 42, 48, 50, 58, 60

Pre-order:48, 40, 35, 42, 58, 50, 60

Post-order: 35, 42, 40, 50, 60, 58, 48

Evaluation scheme: Correct answer will be given full marks and partial marks to be awarded depending on the correctness of the steps.

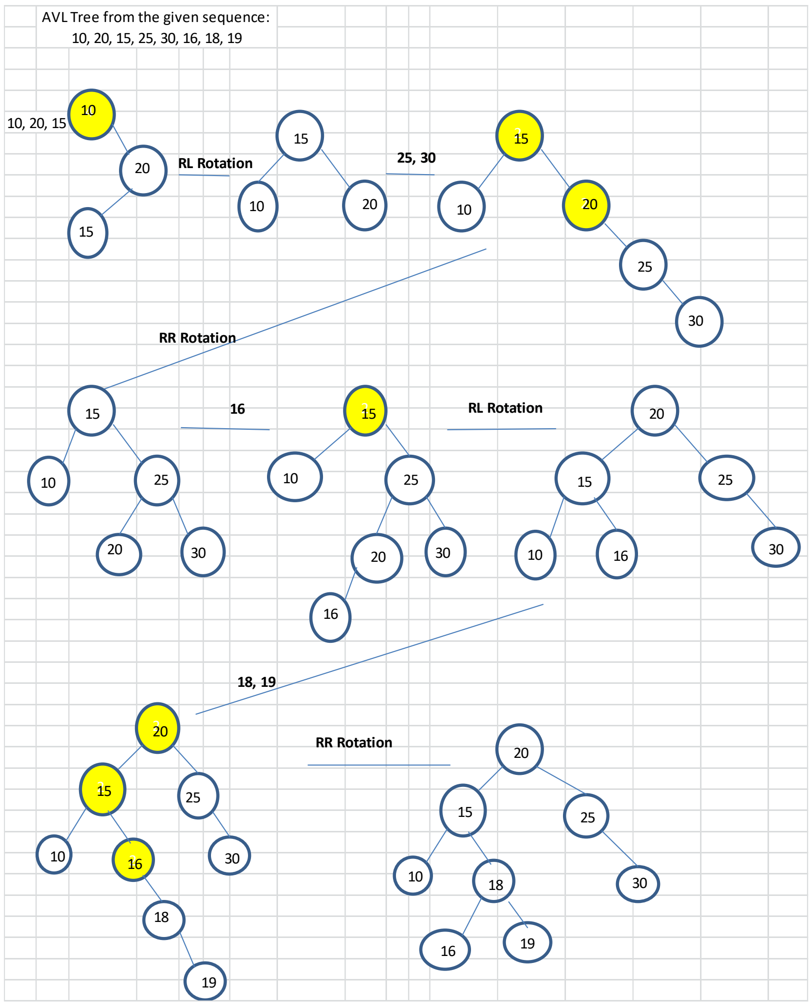
Given two single linked lists, write the code to merge to make one list, taking nodes alternately between the two lists. So the output with {1, 2, 3} and {7, 13, 1} should yield {1, 7, 2, 13, 3, 1}. If either list runs out, all the nodes should be taken from the other list. [4]

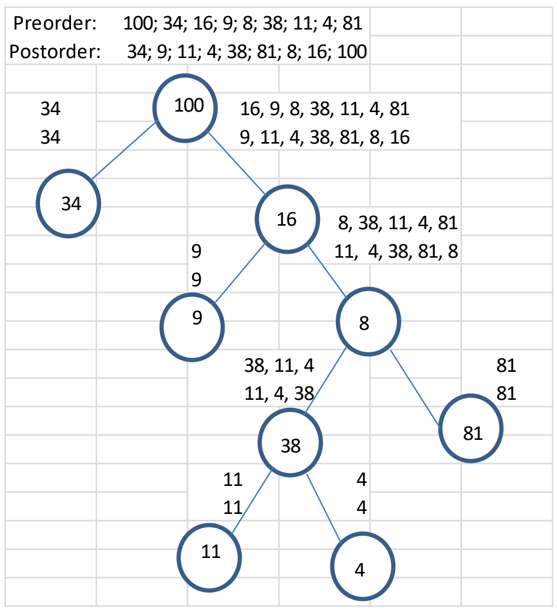
Sol:

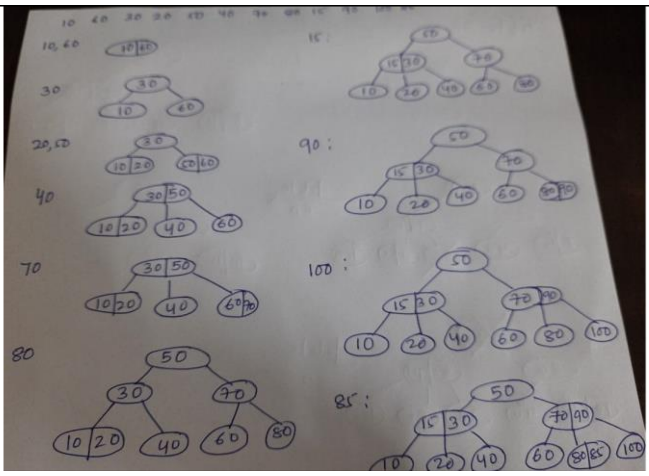
```
/* Given two single linked lists, write the code to merge to make one list, taking
nodes alternately between the two lists.
 So the output with {1, 2, 3} and {7, 13, 1} should yield {1, 7, 2, 13, 3, 1}. If
either list runs out,
all the nodes should be taken from the other list.*/

#include <stdio.h>
#include <stdlib.h>


struct Node
{
        int data;
        struct Node *next;
};

void push(struct Node ** head_ref, int new_data)
{
        struct Node* new_node =
                (struct Node*) malloc(sizeof(struct Node));
        new_node->data = new_data;
        new_node->next = (*head_ref);
        (*head_ref) = new_node;
}

void printList(struct Node *head)
{
        struct Node *temp = head;
        while (temp != NULL)
        {
                printf("%d ", temp->data);
                temp = temp->next;
        }
        printf("\n");
}

void merge(struct Node *p, struct Node **q)
{
        struct Node *p_curr = p, *q_curr = *q;
        struct Node *p_next, *q_next;

        while (p_curr != NULL && q_curr != NULL)
        {
                p_next = p_curr->next;
                q_next = q_curr->next;


                q_curr->next = p_next;
                p_curr->next = q_curr;


                p_curr = p_next;
                q_curr = q_next;
        }

        *q = q_curr;
}

int main()
{
        struct Node *p = NULL, *q = NULL;
        push(&p, 3);
        push(&p, 2);
        push(&p, 1);
        printf("First Linked List:\n");
        printList(p);

        push(&q, 8);
        push(&q, 7);
        push(&q, 6);
        push(&q, 5);
        push(&q, 4);
        printf("Second Linked List:\n");
        printList(q);

        merge(p, &q);

        printf("Modified First Linked List:\n");
        printList(p);

        printf("Modified Second Linked List:\n");
        printList(q);

        getchar();
        return 0;
}
```

Construct an AVL tree by inserting the following elements in the order of their occurrence. Clearly mention which rotations you are applying in each step. 10, 20, 15, 25, 30, 16, 18, 19. [5]

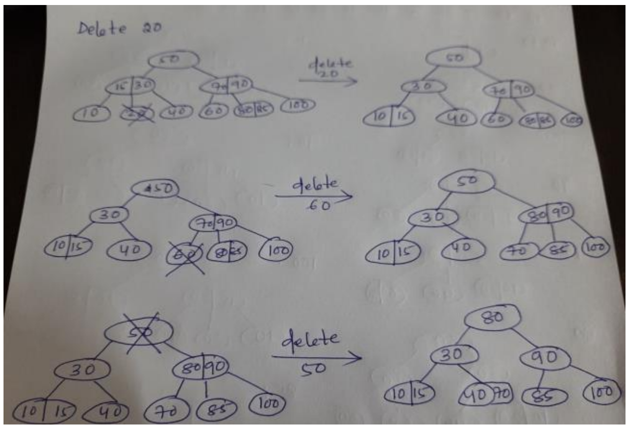
Sol:



	Write the INORDER, PREORDER, and POSTORDER traversal of the above AVL tree. [3] Sol: In-order:10, 15, 16, 18, 19, 20, 25, 30 Pre-order:20, 15,10, 18, 16, 19, 25, 30 Post-order: 10, 16, 19, 18, 15, 30, 25, 20	
Q.No: 10	**Evaluation scheme: Correct answer will be given full marks and partial marks to be awarded depending on the correctness of the steps.** Suppose we know the preorder and postorder traversal sequences of a binary tree T. Can we uniquely determine the binary tree? Answer with a short justification. [2] Sol: No. Binary tree cannot be uniquely determined by its Pre order and post order traversal. If any internal node is having only left or right child, it cannot be determined through the above two order traversal. So unique tree cannot be drawn for these two traversal Let the preorder traversal sequence of T be 100; 34; 16; 9; 8; 38; 11; 4; 81 and postorder traversal sequence be 34; 9; 11; 4; 38; 81; 8; 16; 100. If all the non-leaf nodes of T have two children, identify T. [4] .Sol: The Tree is:  Preorder: 100; 34; 16; 9; 8; 38; 11; 4; 81 Postorder: 34; 9; 11; 4; 38; 81; 8; 16; 100 34 34 100 16, 9, 8, 38, 11, 4, 81 9, 11, 4, 38, 81, 8, 16 34 16 8, 38, 11, 4, 81 11, 4, 38, 81, 8 9 9 9 8 38, 11, 4 11, 4, 38 81 81 38 81 11 11 4 4 11 4 Illustrate the construction of 3-way B-Tree from the following sequence of elements. [3+3] 10, 60, 30, 20, 50, 40, 70, 80, 15, 90, 100, 85. Delete 20, 60 and 50 from the constructed B-tree with proper clarification . Sol: Create B Tree of order 3:	CO-4, CO-6



Deletion from a B tree:



Evaluation scheme: Correct answer will be given full marks and partial marks to be awarded depending on the correctness of the steps.

Given a Binary Search Tree (BST) and a range low to high (inclusive). Write a non recursive code to count the number of nodes in the BST that lie in the given range. [4]

Input:

```
      10
     /  \
    5    50
   /    /   \
  1   40     100
```

low = 5, high = 45

Output: 3 (5, 10, 40, are the node in the range.)

Sol: struct Node

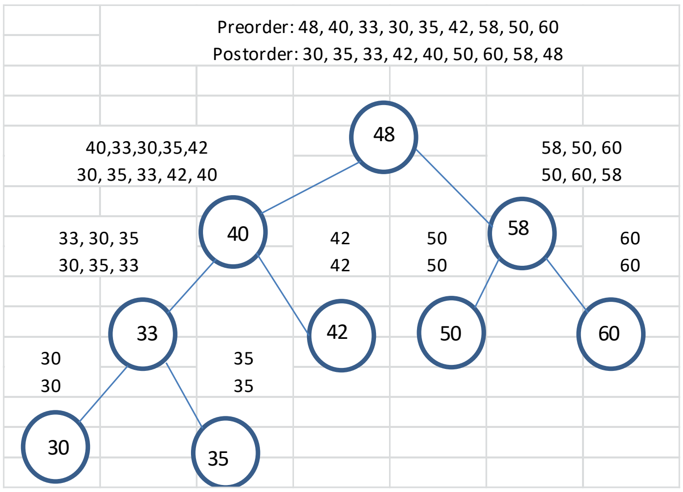
```
{
  int info;
  Struct Node *left, *right;
};
```

```
int getCount(struct Node* node)
{
  if(!node)
  return 0;

  /*Assume a queue named queue to be implemented which has enqueue() and
dequeue() function and is used in level order traversal of BST*/

int count=0;
enqueue(node);
while(!isQueueEmpty())
{
 //function peek returns the element at front position in the queue
struct node *temp=peek();
dequeue();
if(temp->data >= low || temp->data < =high)
count++;

if(temp->left->data>=low || temp->left->data<=high)
push(temp->left);

if(temp->right->data >=low || temp->right->data<=high)
push(temp->right);
}

}
```

Let the preorder traversal sequence of a Tree be 48, 40, 33, 30, 35, 42, 58, 50, 60 and postorder traversal sequence be 30, 35, 33, 42, 40, 50, 60, 58, 48. If all the non-leaf nodes of the Tree have two children, identify the Tree and find the inorder traversal of it. [4]

Sol:

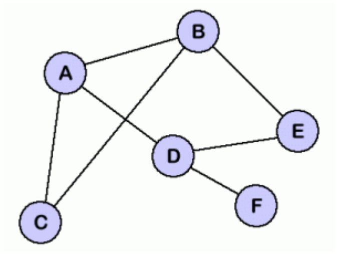
The Tree is:



In order traversal is: 30, 33, 35, 40, 42, 48, 50, 58, 60

Write a procedure to find approachable nodes from a given source in a graph using stack as an intermediate data structure. Apply the above procedure to find the series of nodes which are approachable from node B

[4]



Sol:

DFS Pseudo code:

Series approachable from node B:

BEDFCA

BCEDFA

BEDFAC

BADFEC

BCADFE

BADFCE

Evaluation scheme: Correct answer will be given full marks and partial marks to be awarded depending on the correctness of the steps.

Write a non-recursive code to find the average of minimum and maximum present in a binary search tree. [2]

Sol:

```
main()
{
struct Tree *T = THeader;
int min, max;
float avg;
if(T == NULL){
    printf("Empty Tree\n");
    return;
}
while(T->left != NULL)
          T = T->left;
min = T->data;
while(T->right != NULL)
          T = T->right;
max = T->data;
avg = (min + max)/2.0;
printf("Average of minimum and maximum values is %f\n",avg);
}
```

Given a set of inserting value 10 through 16 (7 continuous values). [2]

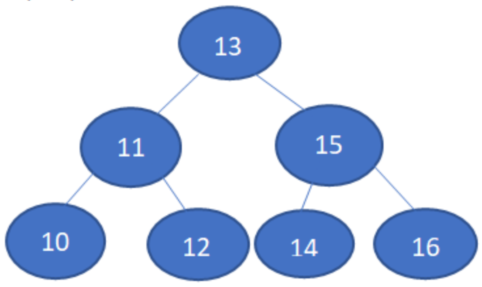
i. Write the sequence of values to be inserted in the BST such that the resulting tree will be a left-skewed tree.With diagram justify your answer.

ii. Write the sequence of values to be inserted in the BST such that the resulting tree will be a complete binary search tree. With diagram justify your answer.

Sol:

i. 16, 15, 14, 13, 12, 11, 10

ii. 13, 11, 15, 10, 12, 14, 16



What is strictly binary tree. Write a code to convert one binary search tree to strictly binary tree by adding the absent child node (left/right) as follows. [4]

i. If the left child of a node is absent then insert a new left child with half valued info of the node's info.
ii. If the right child of a node is absent then insert a new right child with double valued info of the node's info.

Sol:
<u>Strictly Binary Tree</u>: A binary tree is said to be a Strictly Binary Tree (or extended binary tree or 2-tree) if each node in the tree has *either no child or exactly two child nodes*.

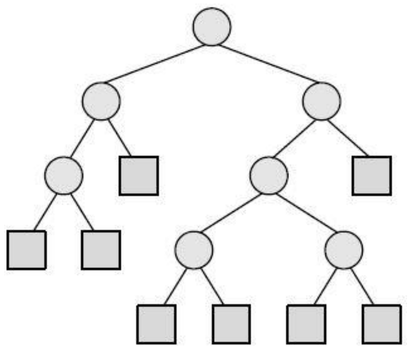


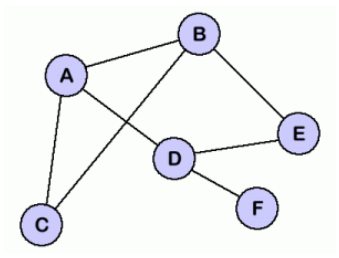
Fig: Strictly Binary Tree

<u>Conversion of Binary Search Tree to Strictly Binary Tree</u>:

```
#include <stdio.h>
#include <stdlib.h>

 #define NODECOUNT 7

 struct bstNode {
       int data;
       struct bstNode *lchild, *rchild;
 };
```

```
  struct bstNode *root = NULL;
  int bstData[] = {100, 80, 120, 70, 90, 110, 130};
  int count = 0;

/* Construct Binary Search Tree from Arrays */

struct bstNode * implementBSTtree(int n)
{
     struct bstNode *newnode;

       if (n >= NODECOUNT)
               return NULL;

       newnode = (struct bstNode *)malloc(sizeof (struct bstNode));

/* Node at position n - have its left child at the position (2 * n) + 1 */
       newnode->lchild = implementBSTtree((2 * n) + 1);
       newnode->data   = bstData[n];

 /* Node at position n - have right child at the position (2 * n) + 2 */
       newnode->rchild = implementBSTtree((2 * n) + 2);
       return newnode;


/* Left child node will be half valued of it's parent node, if only right child present*/
if (newnode->lchild == NULL && newnode->rchild != NULL)
newnode->lchild = bstData[n] / 2;

/* Right child node will be double valued of it's parent node, if only left child present*/
if (newnode->lchild != NULL && newnode->rchild == NULL)
newnode->rchild = 2 * bstData[n];

}



int main() {
       int i = 0;
       printf("Data in Array:\n");
       while (i < NODECOUNT) {
               printf("%d ", bstData[i]);
               i++;
       }
       i = 0;
       root = implementBSTtree(i);
       printf("\n");
       return 0;
  }
```

Write a procedure to find approachable nodes from a given source in a graph using queue as an intermediate data structure. Apply the above procedure to find the series of nodes which are approachable from node B [4]



Sol:

<u>BFS Pseudo code:</u>

Series approachable from node B:

BACEDF
BAECDF
BCADEF
BCEADF
BEACDF
BECADF

Q.No: 11

Evaluation scheme: Correct answer will be given full marks and partial marks to be awarded depending on the correctness of the steps.

CO-5

Illustrate the steps to sort the numbers (in ascending order) 9, 2, 5, 6, 1, 4, 8 by using quicksort algorithm. [4]

Sol:

Pivot						i<j
9	2	5	6	1	4	8
i						j

Pivot						i<j
9	2	5	6	1	4	8
	i					j

Pivot						i<j
9	2	5	6	1	4	8
		i				j

Pivot						i<j
9	2	5	6	1	4	8
			i			j

Pivot						i<j
9	2	5	6	1	4	8
	i			i		j

Pivot						
9	2	5	6	1	4	8
					i	j

Pivot						i<=j
9	2	5	6	1	4	8
						j, i

Pivot						i>last index
9	2	5	6	1	4	8
j	i					j

Pivot				i>j, swap A[pivot] with A[j]		
9	2	5	6	1	4	8
						j

8	2	5	6	1	4	9

pivot					
8	2	5	6	1	4
i					j

pivot					
8	2	5	6	1	4
	i				j

pivot					
8	2	5	6	1	4
		i			j

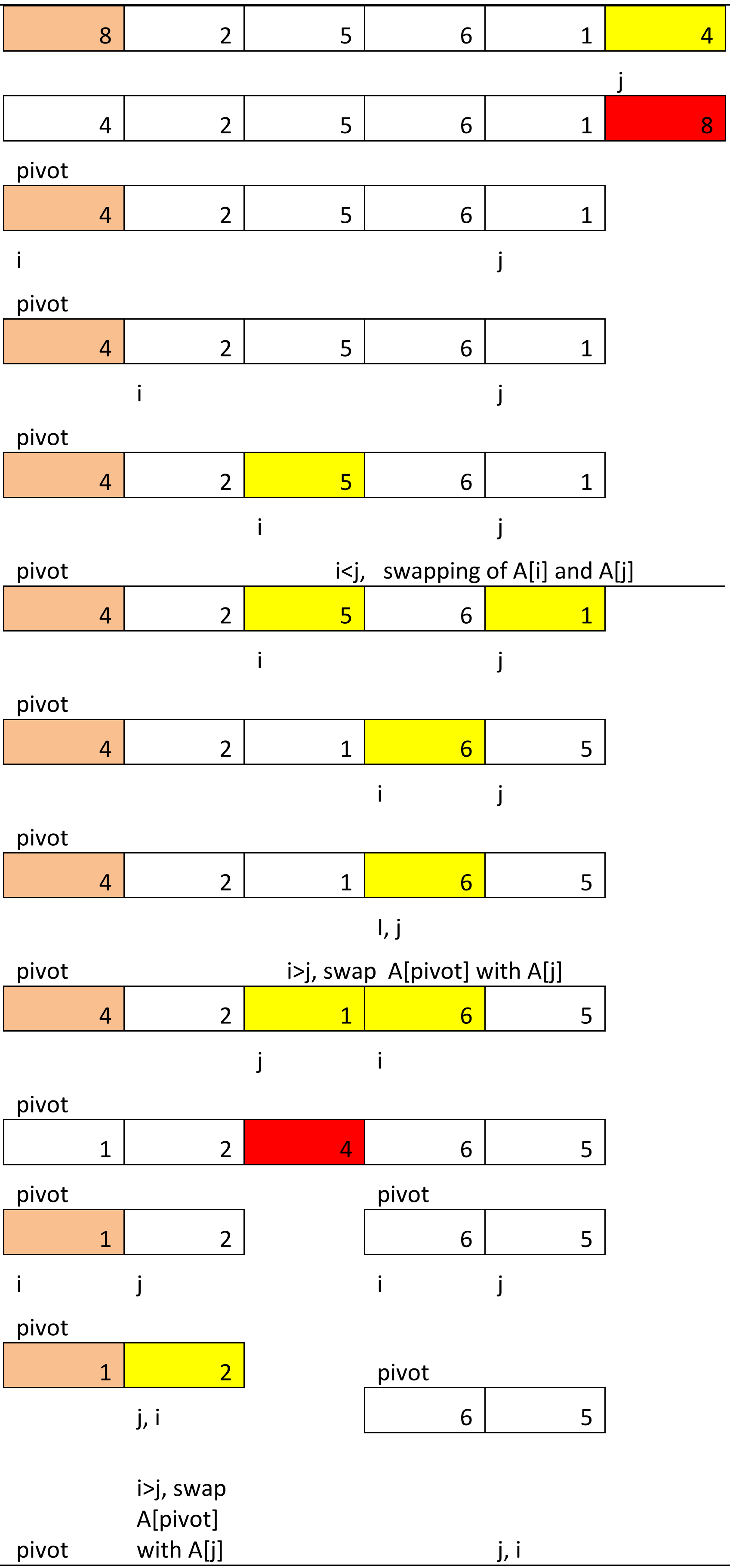
pivot					
8	2	5	6	1	4
			i		j

pivot					
8	2	5	6	1	4
				i	j

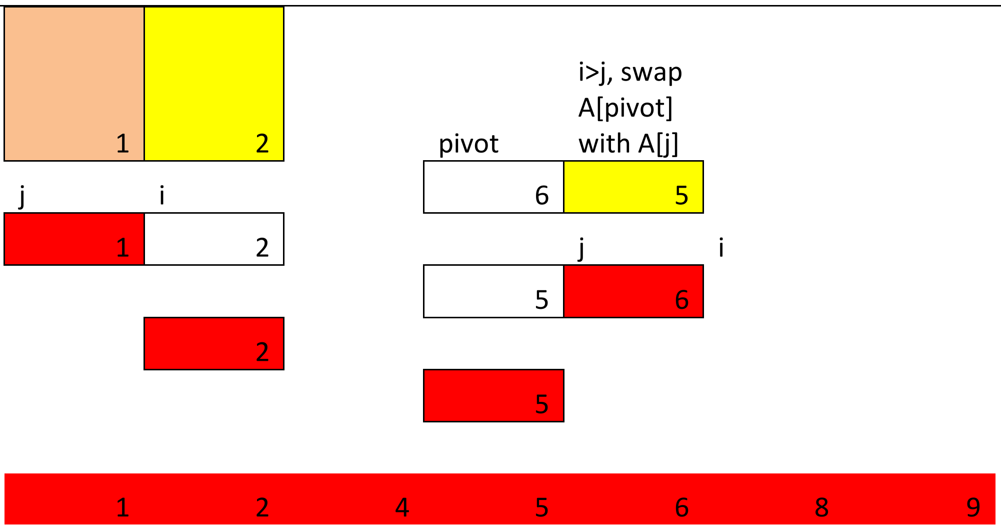
pivot					
8	2	5	6	1	4
					j, i

pivot					i>last index	
8	2	5	6	1	4	
					j	i

pivot   i>j, swap A[pivot] with A[j]

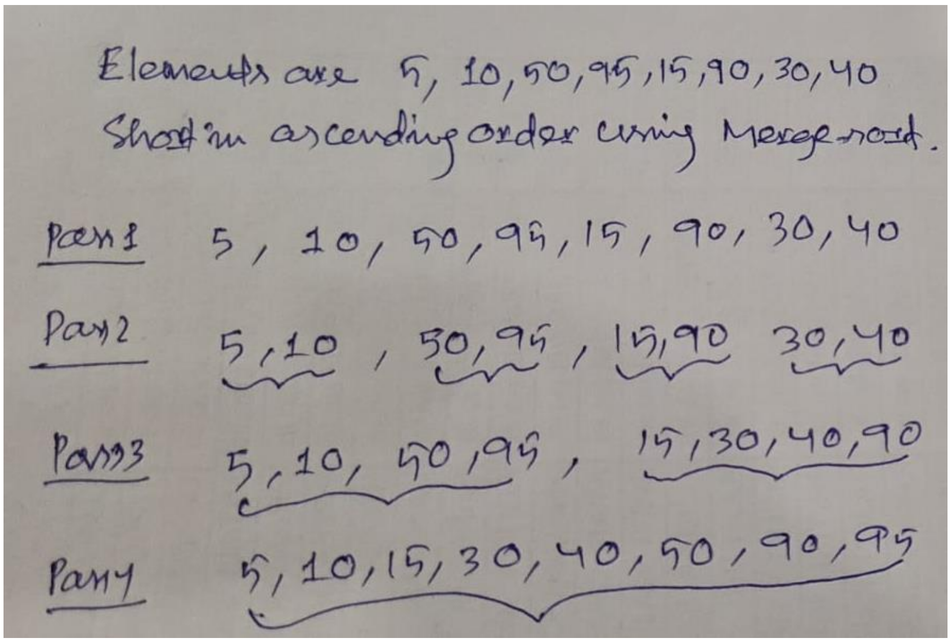


8 2 5 6 1 4
j i
4 2 5 6 1 8
pivot
4 2 5 6 1
i j
pivot
4 2 5 6 1
i j
pivot
4 2 5 6 1
i j
pivot
i<j, swapping of A[i] and A[j]
4 2 5 6 1
i j
pivot
4 2 1 6 5
i j
pivot
4 2 1 6 5
I, j
pivot
i>j, swap A[pivot] with A[j]
4 2 1 6 5
j i
pivot
1 2 4 6 5
pivot
1 2
pivot
6 5
i j
i j
pivot
1 2
j, i
pivot
6 5
i>j, swap A[pivot] with A[j]
pivot
j, i



Show the step-by-step process to arrange the sequence of elements 5, 10, 50, 95, 15, 90, 30, 40 in ascending order using merge sort. [4]

Sol:



Elements are 5, 10, 50, 95, 15, 90, 30, 40

Short in ascending order using Merge sort.

Pass 1 5, 10, 50, 95, 15, 90, 30, 40

Pass 2 5, 10 , 50, 95 , 15, 90 30, 40

Pass 3 5, 10, 50, 95 , 15, 30, 40, 90

Pass 4 5, 10, 15, 30, 40, 50, 90, 95

How many collisions occur if the hash addresses are generated using the modulo division method, where the table size is 64.

9893, 2341, 4312, 7893, 4531, 8731, 3184. [4]

Sol:

With modulo 64:

9893%64=37

2341%64=37 (one collision)

4312%64=24

7893%64=21

4531%64=51

8731%64=27

3184%64=48

Evaluation scheme: Correct answer will be given full marks and partial marks to be awarded depending on the correctness of the steps.

An array contains the following elements: 27, 30, 34, 45, 56, 59, 61. Show the step-by-step process to search 61 using binary search. Show the values of START, MID, and END in each step. [4]

Key = 61

	0	**1**	**2**	**3**	**4**	**5**	**6**
Step1	27	30	34	45	56	59	61

Start=0, end= 6
Here the mid element is 3. Since key≠ 45 and key > 45, the right half is considered.

Step 2

56	59	61

Start= 4, end= 6
Here the mid element is 59. Since key≠ 59 and key > 59, the right half is considered.

Step3

61

Start= 6, end= 6
Now the only element is the key (61), we got the key .

Illustrate the steps to sort the numbers (in ascending order) 47, 21, 23, 56, 12, 87, 19 by using quicksort algorithm. [4]
Sol:

Pivot						i<j
47	21	23	56	12	87	19
i						j

Pivot						i<j
47	21	23	56	12	87	19
	i					j

Pivot						i<j
47	21	23	56	12	87	19
		i				j

Pivot						i<j
47	21	23	56	12	87	19
			i			j

Pivot						i<j
47	21	23	56	12	87	19
			i			j

Pivot i<j, swapping of A[i] and A[j]

47	21	23	19	12	87	56
			i			j

Pivot i<j

47	21	23	19	12	87	56
				i		j

Pivot i<j

47	21	23	19	12	87	56
					i	j

Pivot i<j

47	21	23	19	12	87	56
					i, j	

Pivot i<j

47	21	23	19	12	87	56
				j	i	

Pivot i>j, swap A[pivot] with A[j]

12	21	23	19	47	87	56
				j	i	

12	21	23	19		87	56

partioned the array from jth location

Left Array

pivot i<j

12	21	23	19
i			j

pivot i<j

12	21	23	19
	i		j

pivot i<j

12	21	23	19
	i	j	

pivot i<j

12	21	23	19
	i, j		

pivot i<j

12	21	23	19
j	i		

pivot i>j, swap A[pivot] with A[j]

12	21	23	19
j	i		

pivot i<j

21	23	19

partioned the array from jth location

Right Array

pivot i<j

87	56	
i	j	

pivot i<=j

87	56	
	j,i	

pivot i>last index

87	56	
	j	i

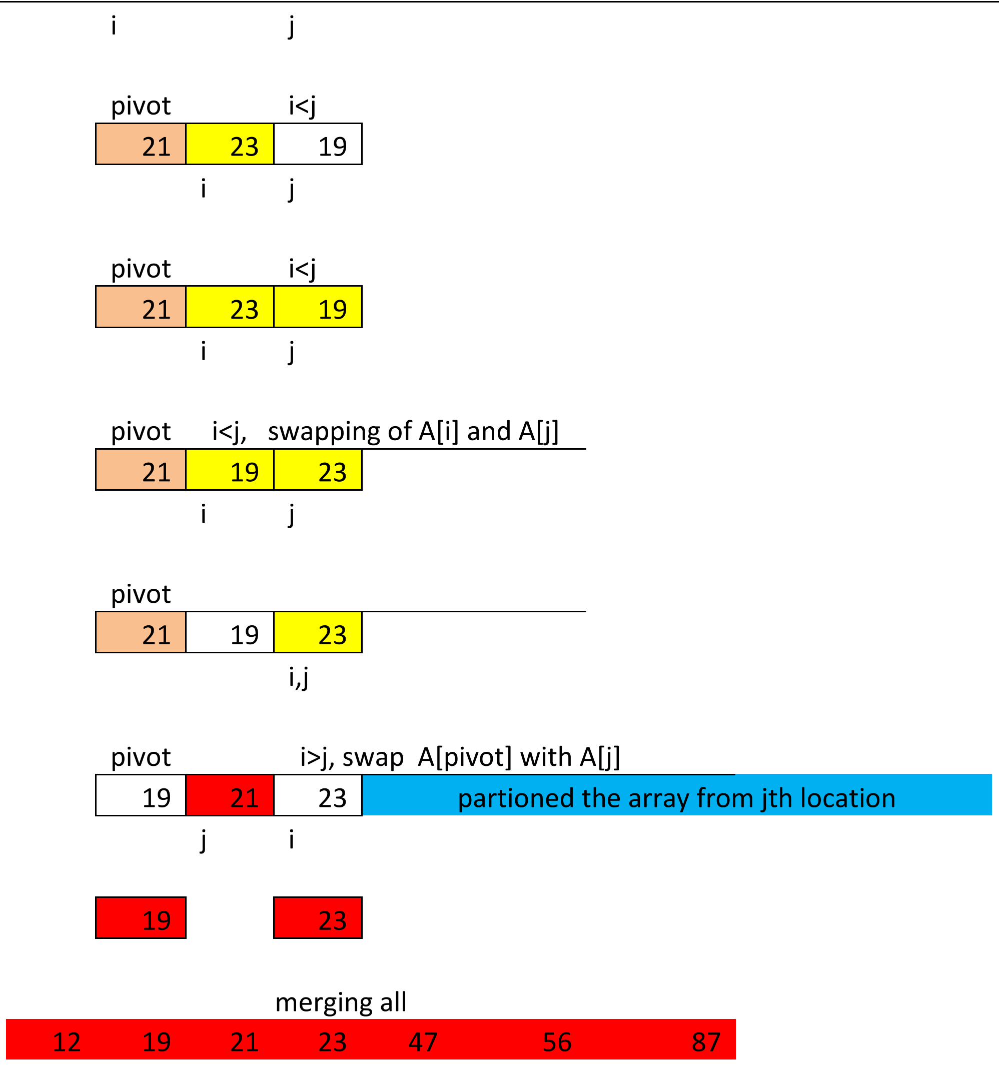
pivot

87	56	
	j	i

pivot i>j, swap A[pivot] with A[j]

56	87	
	j	i

56



How many collisions occur if the hash addresses are generated using the modulo division method, where the table size is 67. [4]

9893, 2341, 4312, 7893, 4531, 8731, 3184.

Sol:
9893%67=44
2341%67=63
4312%67=24
7893%67=54
4531%67=42
8731%67=21
3184%67=35
No collision.

Evaluation scheme: Correct answer will be given full marks and partial marks to be awarded depending on the correctness of the steps.

Illustrate the steps to sort the numbers (in ascending order) 2, 5, 20, 34, 13, 19, 7 by using quicksort algorithm. [4]

Sol:

Pivot i<j

2	5	20	34	13	19	7

i j

Pivot i<j

2	5	20	34	13	19	7

i j

Pivot i<j

2	5	20	34	13	19	7

i j

Pivot i<j

2	5	20	34	13	19	7

i j

Pivot i<j

2	5	20	34	13	19	7

i j

Pivot i<j, swapping of A[i] and A[j]

2	5	20	34	13	19	7

i j

Pivot i<=j

2	5	20	34	13	19	7

i, j

Pivot i>j

2	5	20	34	13	19	7

j i

Pivot i<j

2	5	20	34	13	19	7	partition 1

pivot

5	20	34	13	19	7

i j

pivot

5	20	34	13	19	7

i j

pivot

5	20	34	13	19	7

i j

pivot

5	20	34	13	19	7

i j

pivot

5	20	34	13	19	7

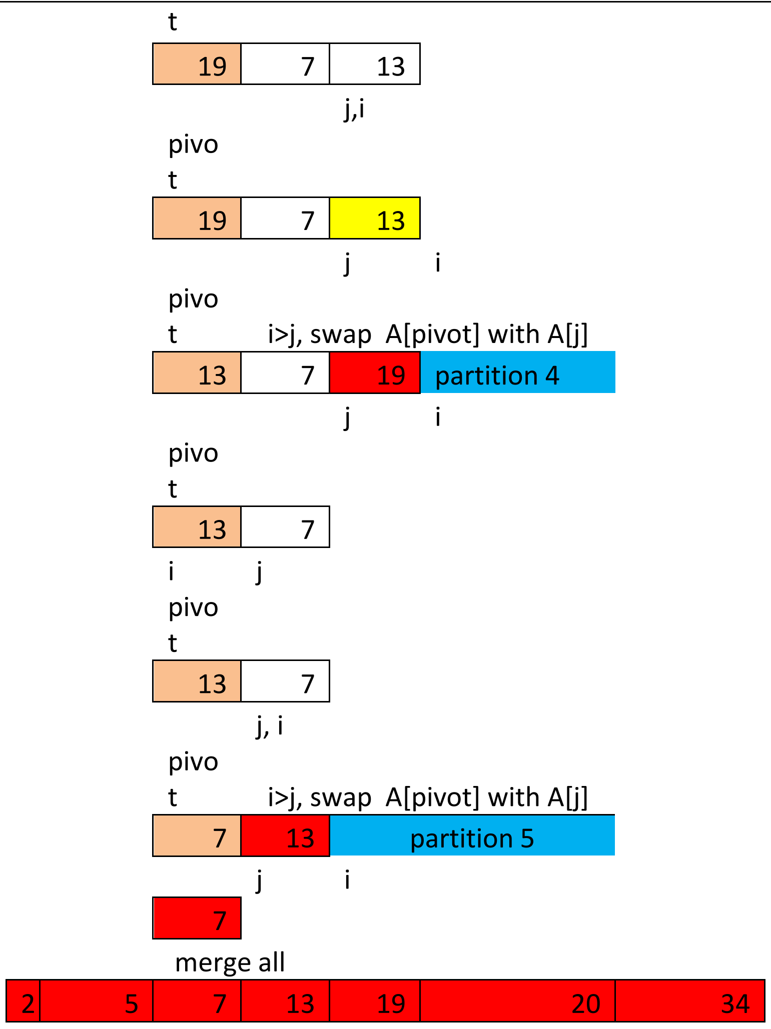
i j

pivot

5	20	34	13	19	7

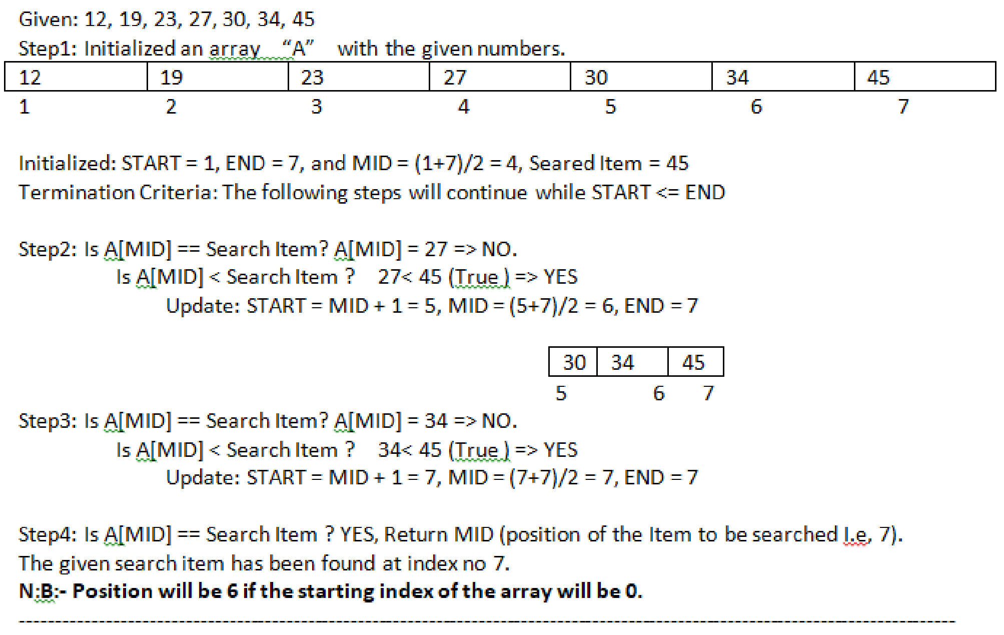
I,j

pivot					i>j	
5	20	34	13	19	7	
j	i					
pivot				i>j, swap A[pivot] with A[j]		
5	20	34	13	19	7	partition 2
j	i					
	pivo					
	t					
	20	34	13	19	7	
	i				j	
	pivo					
	t					
	20	34	13	19	7	
		i			j	
	pivo					
	t	i<j, swap A[i] with A[j]				
	20	7	13	19	34	
		i			j	
	pivo					
	t					
	20	7	13	19	34	
			i		j	
	pivo					
	t					
	20	7	13	19	34	
				i	j	
	pivo					
	t					
	20	7	13	19	34	
					j,i	
	pivo					
	t		i>j, swap A[pivot] with A[j]			
	20	7	13	19	34	
				j	i	
	19	7	13	20	34	partition 3
				j	i	
	pivo					
	t					
	19	7	13		34	
	i		j			
	pivo					
	t					
	19	7	13			
		i	j			
	pivo					



An array contains the following elements: 12, 19, 23, 27, 30, 34, 45. Show the step-by-step process to search 45 using binary search. Show the values of START, MID, and END in each step. [4]

Sol:


Given: 12, 19, 23, 27, 30, 34, 45

Step1: Initialized an array "A" with the given numbers.

12	19	23	27	30	34	45
1	2	3	4	5	6	7

Initialized: START = 1, END = 7, and MID = (1+7)/2 = 4, Seared Item = 45

Termination Criteria: The following steps will continue while START <= END

Step2: Is A[MID] == Search Item? A[MID] = 27 => NO.

Is A[MID] < Search Item ? 27< 45 (True) => YES

Update: START = MID + 1 = 5, MID = (5+7)/2 = 6, END = 7

30	34	45
5	6	7

Step3: Is A[MID] == Search Item? A[MID] = 34 => NO.

Is A[MID] < Search Item ? 34< 45 (True) => YES

Update: START = MID + 1 = 7, MID = (7+7)/2 = 7, END = 7

Step4: Is A[MID] == Search Item ? YES, Return MID (position of the Item to be searched I.e, 7).

The given search item has been found at index no 7.

N:B:- Position will be 6 if the starting index of the array will be 0.

Insert the following keys in an array of size 7 using the modulo division method. If collision occurs, then use linear probing to resolve the collisions. 94, 37, 29, 40, 84, 88, 102. [4]

Sol:

94%7=3

37%7=2

29%7=1

40%7=5

84%7=0

88%7=4

102%7=4(collision, so moved to index 6)

Index	Key
0	84
1	29
2	37
3	94
4	88
5	40
6	102