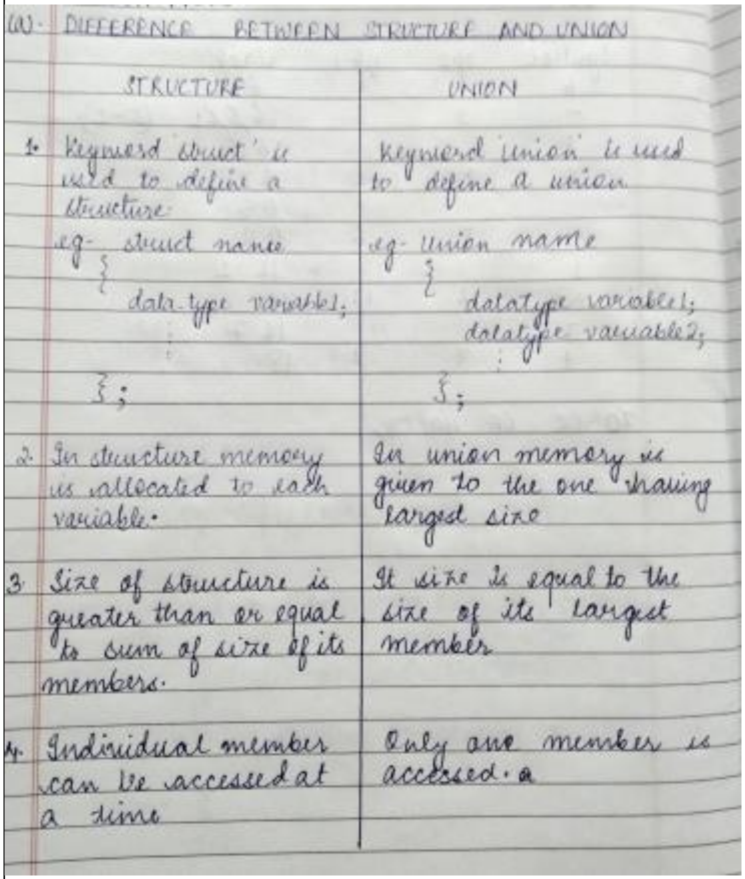
Internal Assessment Test 2 – Oct-2020								
Sub:	Data Structures and Applications				Sub Code:	18CS32	Branch:	CSE
Date:	29/10/2020	Duration:	90 mins	Max Marks:	50	Sem/ Sec:	3rd /ABC	OBE

Answer any FIVE FULL Questions	MARK S	CO	RB T
1. a) Differentiate between Structure and Union.?	[4]	CO1	L2



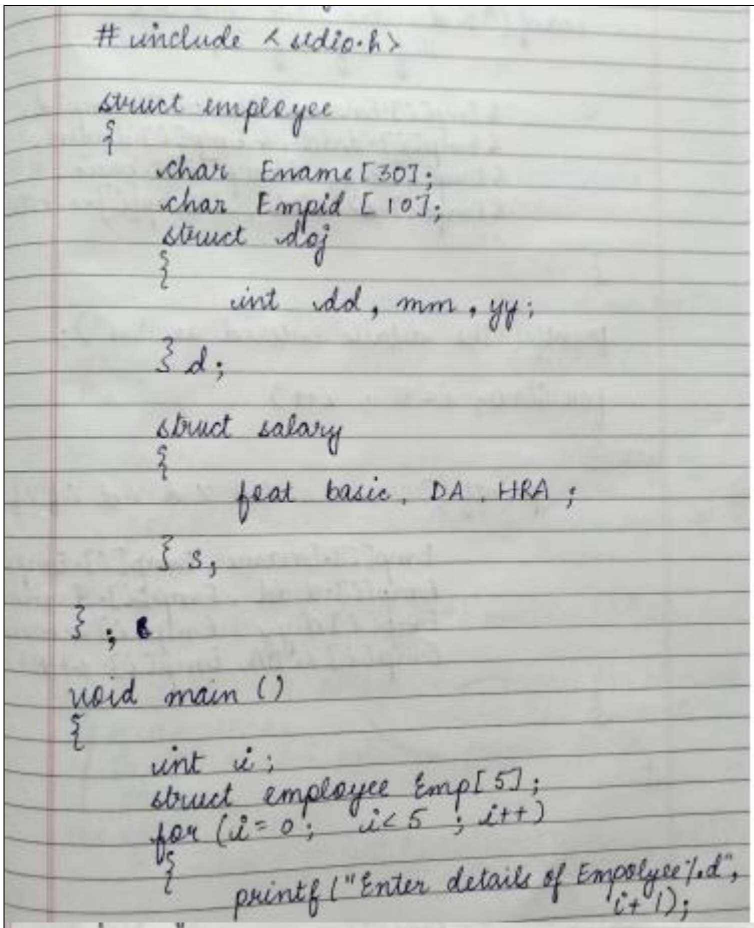
(a). DIFFERENCE BETWEEN STRUCTURE AND UNION

	STRUCTURE	UNION

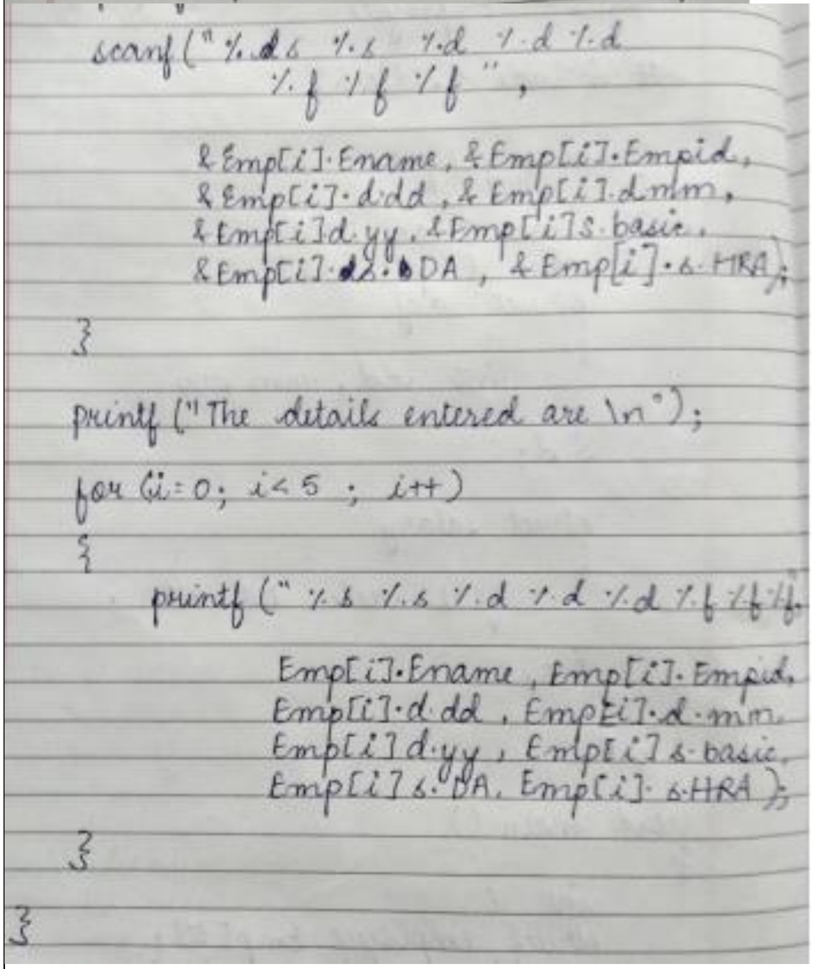
1.	Keyword 'struct' is used to define a structure
	eg- struct name
	{
	data type variable1;
	:
	};
		Keyword 'union' is used to define a union
		eg- union name
		{
		datatype variable1;
		datatype variable2;
		:
		};

2.	In structure memory is allocated to each variable.	In union memory is given to the one having largest size
3	Size of structure is greater than or equal to sum of size of its members.	It size is equal to the size of its largest member
4.	Individual member can be accessed at a time	Only one member is accessed. a

1. b)Write a C program with an appropriate structure definition and variable declaration to read and display information about 5 employees using nested structures. Consider the following fields like Ename, Empid, DOJ (Date, Month, Year) and Salary (Basic, DA, HRA). **Answer:**	[6]	CO1	L3



```
#include <stdio.h>

struct employee
{
    char Ename[30];
    char Empid [10];
    struct doj
    {
        int dd, mm, yy;
    } d;

    struct salary
    {
        feat basic, DA, HRA ;
    } s,
};
void main ()
{
    int i;
    struct employee Emp[5];
    for (i=0;  i<5 ; i++)
    {
        printf ("Enter details of Empolyee%d",
                                        i+1);
```



```
scanf (" %s  %s  %d %d %d
         %f %f %f ",

        &Emp[i].Ename, &Emp[i].Empid,
        &Emp[i].d.dd, &Emp[i].d.mm,
        &Emp[i]d.yy, &Emp[i]s.basic,
        &Emp[i].s.DA , &Emp[i].s.HRA);

}

printf ("The details entered are \n");

for (i=0; i<5 ; i++)
{
    printf (" %s %s %d %d %d %f %f %f.

            Emp[i].Ename, Emp[i].Empid,
            Emp[i].d.dd , Emp[i].d.mm,
            Emp[i]d.yy, Emp[i]s.basic,
            Emp[i]s.DA, Emp[i].s.HRA );
}

}
```

2 **a) Write the output of the following C program: Assume Address of a = 5000, Address of p = 6000, Address of q = 7000.** [2] CO1 L3

```
#include<stdio.h>

int main()
{
int a = 10;
int *p = &a;
int **q = &p;

printf("\n%d", *(&a));
printf("\n%p", &p);
printf("\n%d", q);
printf("\n%p", *q);

    return 0;
}
```

Answer:

```
10
6000
6000
5000
```

2 **b) What is dynamic memory allocation? Explain different functions for allocating and deallocating memory dynamically with syntax and examples.** [8] CO1 L2

Dynamic Memory Allocation can be defined as a procedure in which the size of a data structure (like Array) is changed during the runtime.
C provides some functions to achieve these tasks. There are 4 library functions provided by C defined under **<stdlib.h>** header file to facilitate dynamic memory allocation in C programming. They are:

1. malloc()
2. calloc()
3. free()
4. realloc()

Dynamic memory allocation

Dynamic memory allocation is an aspect of allocating and freeing memory according to your needs. Dynamic memory is managed and served with pointers that point to the newly allocated space of memory in an area which we call the **heap**. Now you can create and destroy an array of elements at runtime without any problems.

To sum up, the automatic memory management uses the stack, and the dynamic memory allocation uses the heap.

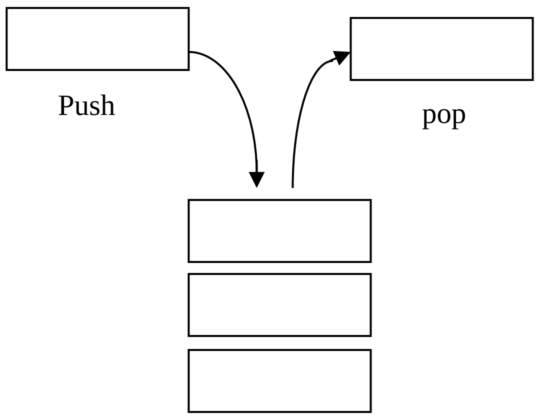
The <stdlib.h> library has functions responsible for dynamic memory management.

Function	Purpose
malloc	Allocates the memory of requested size and returns the pointer to the first byte of allocated space.
calloc	Allocates the space for elements of an array. Initializes the elements to zero and returns a pointer to the memory.
realloc	It is used to modify the size of previously allocated memory space.
Free	Frees or empties the previously allocated memory space.

Example and syntax for each

3a. **Define Stack. Explain the different operations you can perform on a Stack with examples.**	[5]	CO3	L2

A stack is a data structure which is an ordered collection of homogeneous data elements where the insertion and deletion operations occur at only one end. This end is often known as top of stack. Here the last element inserted will be on the top of the stack. Since deletion is done from the same end, last element inserted will be the first element to be removed out from the stack and so on. Thus the stack is also called as Last – In – First – Out (LIFO).



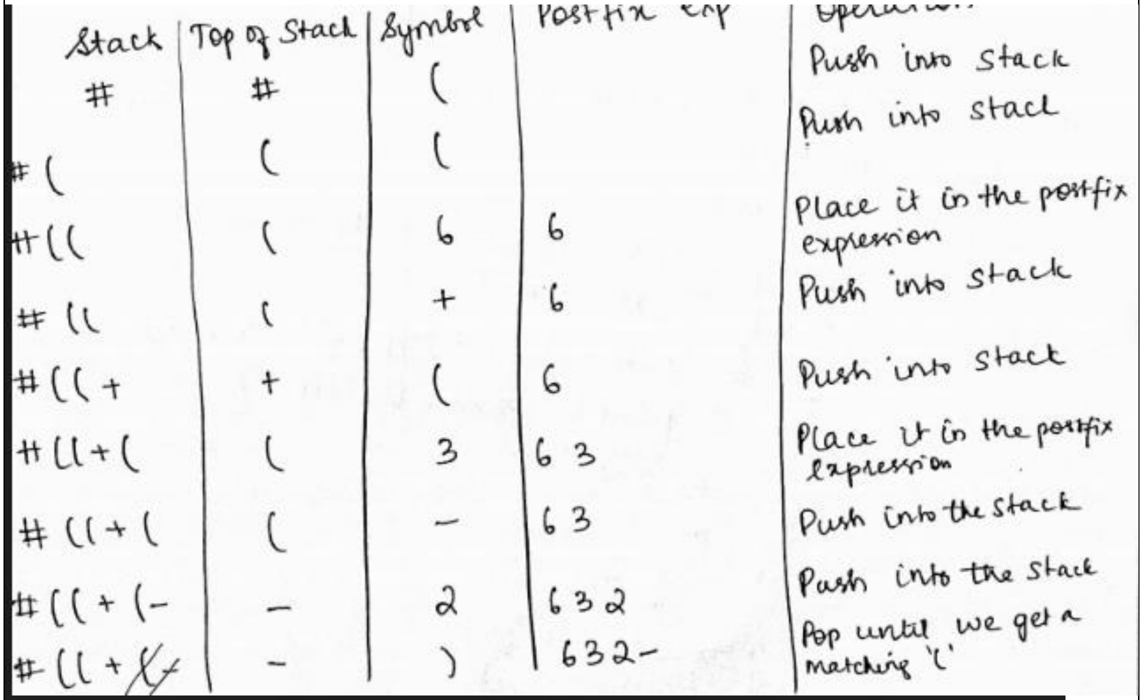
The stack concept can be compared in analogy to a stack of plates in a cafeteria where ever every new plate added is at the top. Similarly every new plate taken off the stack is also from the top of the stack. That is the addition of the plates is at the top and removal of the plates is also at the top. This concept illustrates the stack.

<u>Operations on stack:</u>

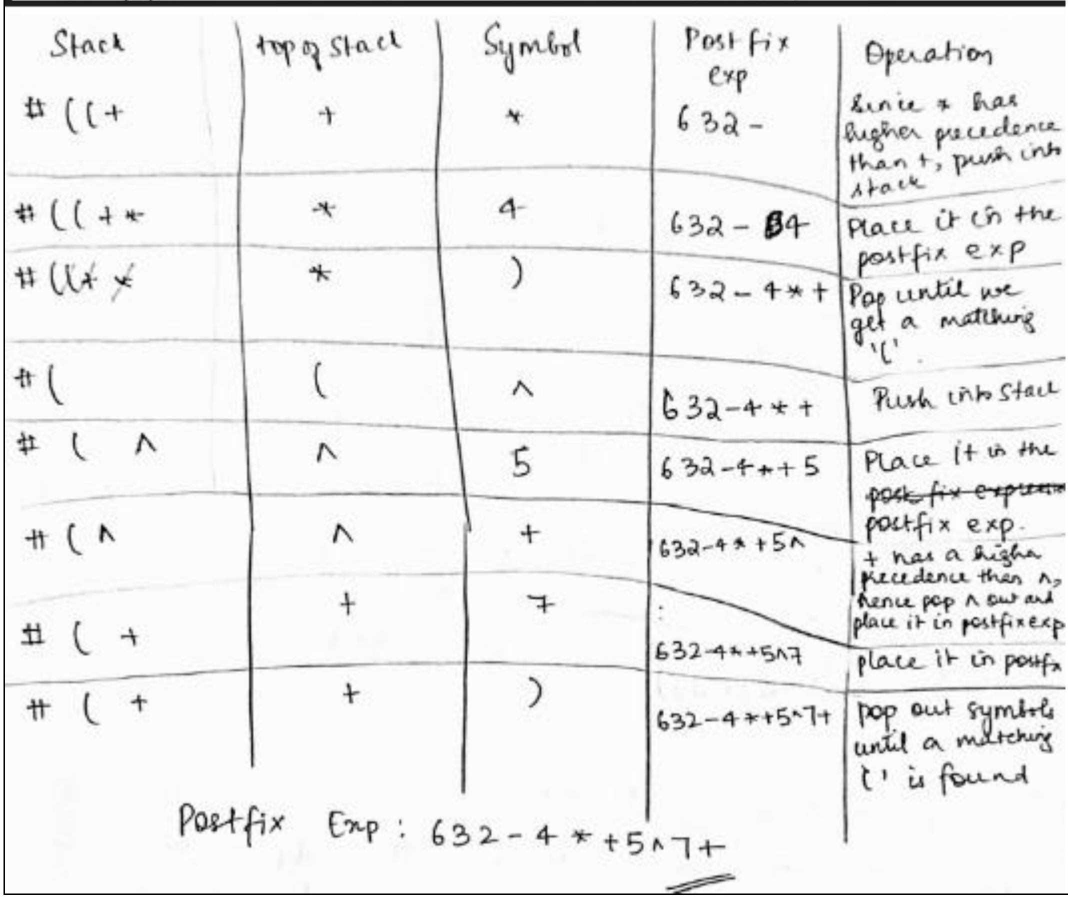
A stack generally has two basic operations such as PUSH and POP

PUSH:	Allows adding an element at the top of the stack
POP:	Allows removing an element from the top of the stack.

3b. Convert the following infix expression to postfix expression: **((6+(3-2)*4)^5+7)**	5	CO3	L3



Stack	Top of Stack	Symbol	Postfix exp	Operation
#	#	(		Push into stack
#(	(	(		Push into stack
#((	(	6	6	Place it in the postfix expression
#((	(	+	6	Push into stack
#((+	+	(	6	Push into stack
#((+(	(	3	6 3	Place it in the postfix expression
#((+(	(	–	6 3	Push into the stack
#((+(–	–	2	6 3 2	Push into the stack
#((+~~(–~~	–	)	632–	Pop until we get a matching '('



Stack	top of Stack	Symbol	Postfix exp	Operation
#((+	+	*	632–	Since * has higher precedence than +, push into stack
#((+*	*	4	632–~~3~~4	Place it in the postfix exp
#((+~~*~~	*	)	632–4*+	Pop until we get a matching '('.
#(	(	^	632–4*+	Push into Stack
#(^	^	5	632–4*+5	Place it in the ~~post fix expression~~ postfix exp.
#(^	^	+	632–4*+5^	+ has a higher precedence than ^, hence pop ^ out and place it in postfix exp
#(+	+	7	632–4*+5^7	place it in postfix
#(+	+	)	632–4*+5^7+	pop out symbols until a matching '(' is found

Postfix Exp : 632–4*+5^7+

4.a **Write C functions for the following on an array of integers with size MAX=5:** **i. Insert Rear on Circular Queue** **ii. Insert Front on Double Ended Queue** iii. **Delete Rear on Normal Queue**	**2+2+2**	CO3	L3

Answer:

Insert rear on circular queue

```
char cq[MAX];
int f=-1, r=-1;

void insert(char ele)
{
   if(f==((r+1)%MAX))
   {
       printf("\nQueue Overflow!!!\n");
       return;
   }

   if(r==-1)
   {


       f=0;
    }

    r=(r+1)%MAX;
    cq[r] = ele;
 }
```

Insert front on double ended queue

```
void insert_front(int ele)
{
    if(f==-1)
    {
      f=r=0;
      q[f]=ele;
    }
    else if(f!=0)
    {
      f--;
      q[f]=ele;
    }
    else
    {
      printf("\nQueue Overflow!!!\n");
    }
}
```

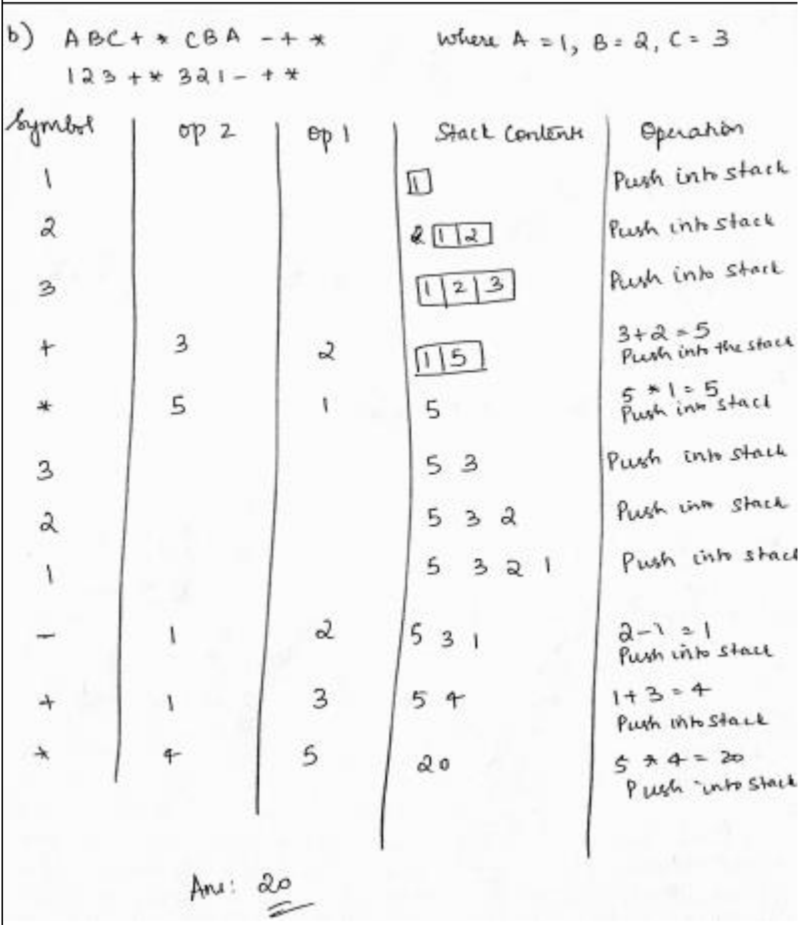
Delete rear on normal queue

```
int delete()
{
    int del;

    if(f == -1) //Queue is Empty.
    {
        printf("\nQueue Underflow!!!\n");
        return -1;
    }
    else if(f == r) //One element in the queue.
    {
        del = q[f];



                f = -1;
                r = -1;
                return del;
            }
            else
            {
                del = q[f];
                f++;
                return del;
            }
        }
```

4b **Evaluate the following Postfix/Suffix Expression:**

ABC+*CBA-+* where A = 1, B = 2, C = 3

4 CO3 L3



b) A B C + * C B A - + * where A = 1, B = 2, C = 3

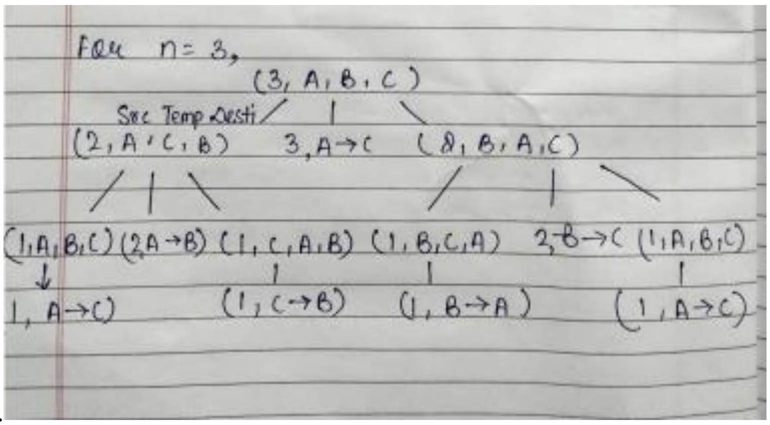
1 2 3 + * 3 2 1 - + *

Symbol	op 2	op 1	Stack Contents	Operation
1			1	Push into stack
2			1 2	Push into stack
3			1 2 3	Push into stack
+	3	2	1 5	3+2 = 5 Push into the stack
*	5	1	5	5 * 1 = 5 Push into stack
3			5 3	Push into stack
2			5 3 2	Push into stack
1			5 3 2 1	Push into stack
-	1	2	5 3 1	2-1 = 1 Push into stack
+	1	3	5 4	1+3 = 4 Push into stack
*	4	5	20	5 * 4 = 20 Push into stack

Ans: 20

5.a Draw a recursive tree for Tower of Hanoi for n = 3 disks

5 CO1 L3



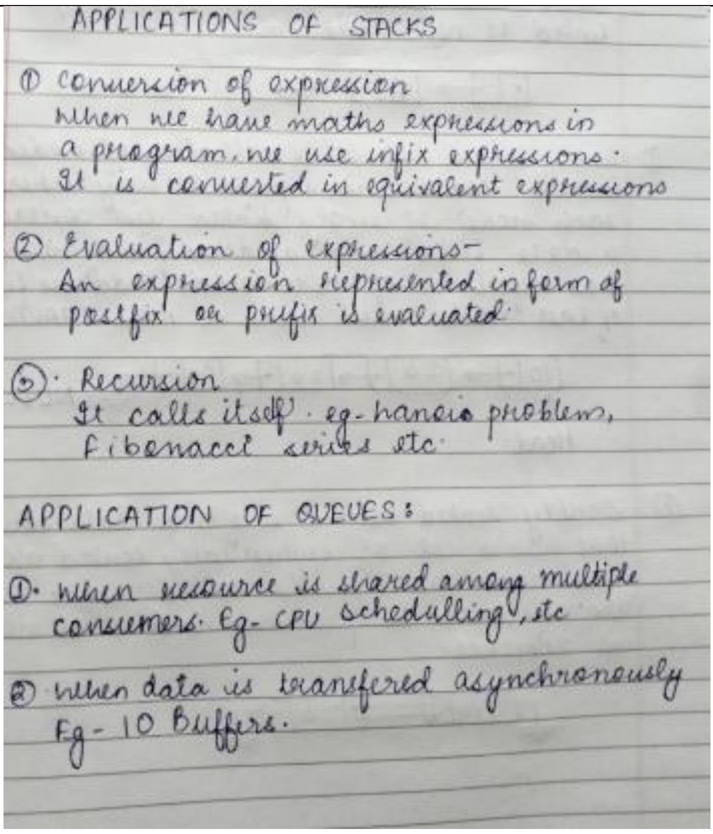
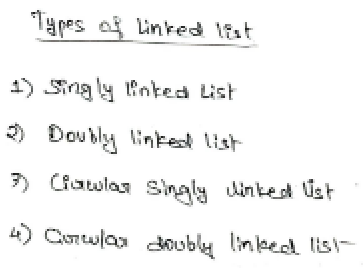
.

5b Give application of Stacks and Queues.

5 CO3 L1

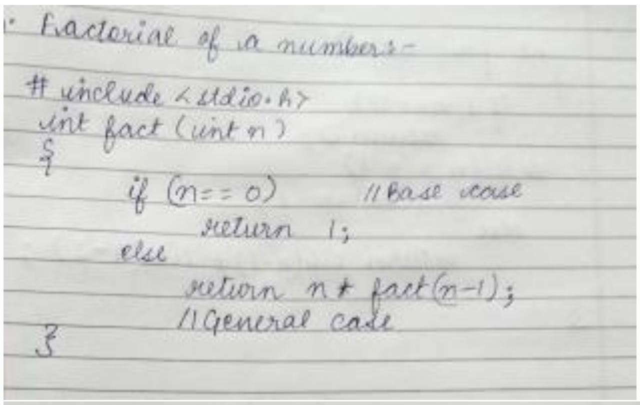
Answer:

5

 APPLICATIONS OF STACKS ① Conversion of expression when we have maths expressions in a program, we use infix expressions. It is converted in equivalent expressions ② Evaluation of expressions– An expression represented in form of postfix or prefix is evaluated. ③. Recursion It calls itself. eg- hanoio problem, Fibonacci series etc. APPLICATION OF QUEUES: ①. when resource is shared among multiple consumers. Eg- CPU schedulling, etc ② when data is transfered asynchronously Eg- IO Buffers.			
6.a Define Linked List. What are the different types of linked list? A linked list is a very common data structure which is an ordered collection of finite, homogeneous data elements called nodes where the linear order is maintained by means of links or pointers. In other words it is used to store similar data in memory. Depending on the requirements the pointers are maintained, and accordingly the linked list can be classified into three major groups: single linked list, double linked list and circular linked list. In a linked list the individual elements are stored somewhere in memory, rather like a family dispersed but still bound together.  Types of Linked list 1) Singly linked List 2) Doubly linked list 3) Circular Singly linked list 4) Circular doubly linked list	4	CO3	L1

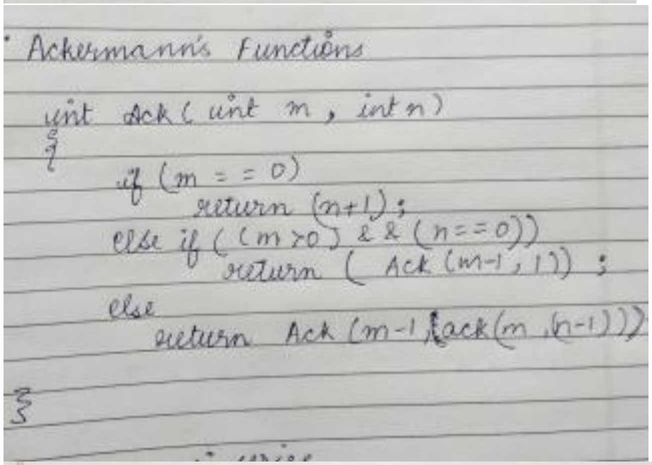
6b. Write a C recursive function for: **i. Factorial of a number** **ii. Ackermann's Function** **iii. Fibonacci Number**	6	CO1	L3

Answer:



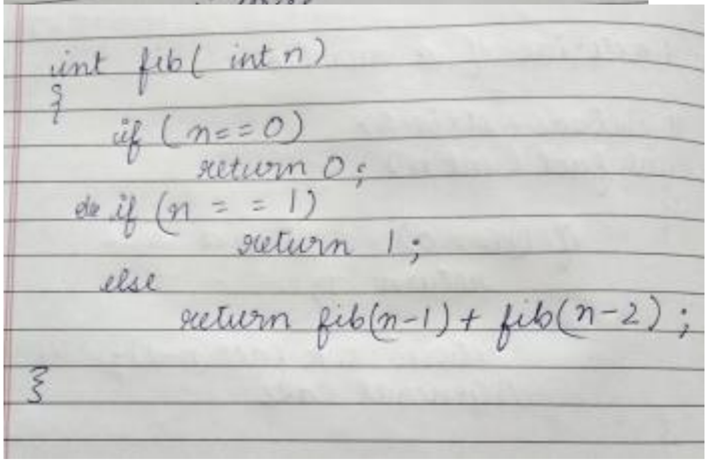
• Factorial of a numbers:-

```
# include <stdio.h>
int fact (int n)
{
      if (n== 0)        //Base case
          return 1;
      else
          return n* fact(n-1);
          //General case
}
```



• Ackermann's Functions

```
int Ack( int m , int n)
{
     if (m = = 0)
          return (n+1);
     else if ( (m>0) && (n==0))
          return ( Ack (m-1, 1)) ;
     else
        return Ack (m-1,ack(m ,n-1)));

}
```



```
int fib( int n)
{
     if ( n==0)
          return 0;
   ~~de~~ if (n = = 1)
          return 1;
     else
          return fib(n-1) + fib(n-2) ;
}
```