USN | | | | | | | | | |

CMRIT

Internal Assessment Test - I

Sub: **DATA STRUCTURES & APPLICATIONS** Code: 15CS33

Date: 20/09/17 Duration: 90 mins Max Marks: 50 Sem: III Branch: ISE

Answer Any **FIVE** Complete Questions.

		Marks	OBE CO	RBT
1	Give the output for the following program. You may represent the memory allocation diagrammatically choosing the memory address you wish. The compiler does not throw any syntax errors.	[10]	CO1	L1

```
#include<stdio.h>
void main()
{
        float b=12;
        float *r,**s;
        int num[5]={3,4,6,2,1};
        int *p=num;
        int *q=num+2;
        r=&b;
        s=&r;
        printf("\n%f\t%d",b,num[2]);
        printf("\n%p",r);
        printf("\n%p",&r);
        printf("\n%p",num);
        printf("\n%d",*(num+2));
        printf("\n%d",*p);
        printf("\n%p",s);
        printf("\n%p",*s);
        printf("\n%p",&num[4]);
        printf("\n%d",*&num[2]);
}
```

2	Write a C program to add two polynomials.	[10]	CO1	L2
3	For the given **sparse matrix and its transpose**, give the triplet representation using one dimensional array, **A** is the given sparse matrix, **B** will be its transpose.	[2+4+4]	CO2	L3

A=

15	0	0	22	0	-15
0	11	3	0	0	0
0	0	0	-6	0	0
0	0	0	0	0	0
91	0	0	0	0	0
0	0	28	0	0	0

4 a) Define stack. Give the C implementation of PUSH and POP functions. Include suitable checks in the function. [6] CO2 L3

b) Consider the following stack of characters, where **STACK** is allocated N=8 memory cells. **ITEM** stores the element deleted from the STACK. [4] CO3 L4

STACK: A,C,D,F,K,___, ___, ___

(For notational convenience, we use "___" to represent an empty memory cell.)

Design the stack as the following operations take place:

a) POP(STACK,ITEM) b) POP(STACK,ITEM)
c) PUSH(STACK,L) d) PUSH(STACK,P)
e) POP(STACK,ITEM) f) PUSH(STACK,R)
g) PUSH(STACK,S) h) POP(STACK,ITEM)

5 Discuss the 'Tower of Hanoi' problem and write a recursive algorithm to solve it. [10] CO4 L3

6 Develop a structure to represent planet in the solar system. Each Planet has fields for the planet's name, its distance from the sun in miles and the number of moons it has. Write a program to read the data for each planet and store. Also print the name of the planet that has less distance from the sun. [10] CO1 L2

OR

Let **S** and **T** be 2 strings: **S = 'I AM PROUD'**
T = 'TO BE AN INDIAN'
i) Find the LENGTH of S and T
ii) Find the SUBSTRING(**S**,4,5) and the SUBSTRING(**T**,10,5)
iii) Find INDEX(**S**,'P') and INDEX(**T**, 'THEN')
iv) DELETE(**'AAABBB'**,3,3)
v) INSERT(**'AAA'**, 2, **'BBB'**)
vi) REPLACE(**'ABABAB'**, **'B'**, **'BAB'**)
vii) Find **S//T**

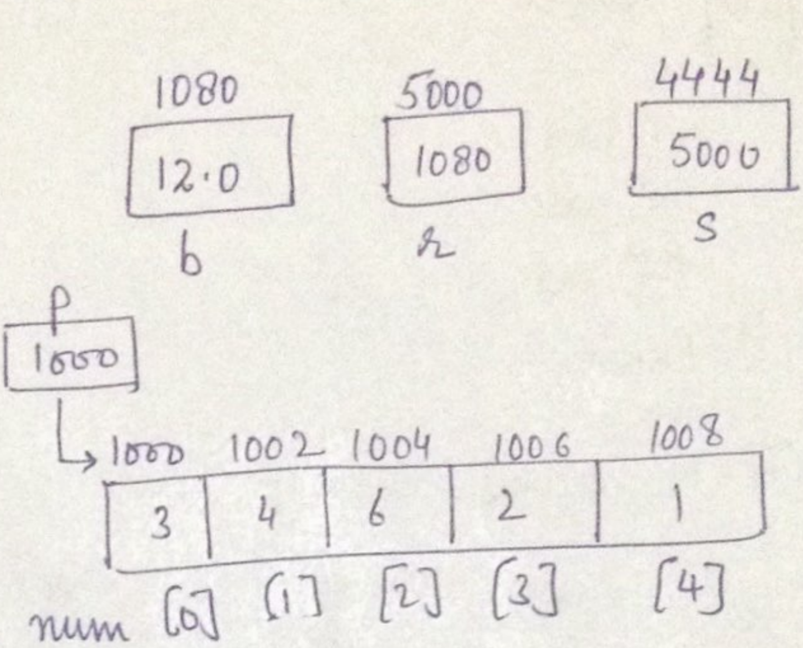
7 Write an algorithm for conversion of an Infix expression to its corresponding Postfix expression. Trace the algorithm and convert the following infix to postfix. [10] CO3 L3

(A+B)*(C^ (D-E)+F)-G

Consider the symbol ^ as exponent.

DSA - IAT I SEPTEMBER 2017-

Q.1 Pointers



1. 12.0 , 6 (b, num[2])
2. 1080 (r)
3. 5000 (&r)
4. 1000 (num)
5. 6 (*(num+2))
6. 3 (*p)
7. 5000 (s)
8. 1080 (*s)
9. 1008 (& num[4])
10. 6 (*& num[2])

r = &b;
s = &r;

Consider int requires 2 bytes (64 bit arch)

Q.2. /* C Prog. to add 2 polynomials */.

↳ Represent polynomials:- Use array of structures.

↳ Addn. of terms (like).

Q3.


```
typedef struct
{
    int row;
    int col;
    int val;
}TERM;

TERM a[100], b[100];
```

	row	col	val
a[0]	6	6	8
a[1]	0	0	15
a[2]	0	3	22
a[3]	0	5	-15
a[4]	1	1	11
a[5]	1	2	3
a[6]	2	3	-6
a[7]	4	0	91
a[8]	5	2	28

$B = A^T =$

	0	1	2	3	4	5
0	15	0	0	0	91	0
1	0	11	0	0	0	0
2	0	3	0	0	0	28
3	22	0	-6	0	0	0
4	0	0	0	0	0	0
5	-15	0	0	0	0	0

	row	col	val
b[0]	6	6	8
b[1]	0	0	15
b[2]	0	4	91
b[3]	1	1	11
b[4]	2	1	3
b[5]	2	5	28
b[6]	3	0	22
b[7]	3	2	-6
b[8]	5	0	-15

Q.4. i) Stack is a data struc. which stores a list of elems. in which an elem. may be inserted or deleted only at one end, called Top of the stack. It follows LIFO.

```
/* PUSH */
void push ()
{
 /* Check if stack is full */
 if (top = = stacksize - 1)
 {
   printf ("stack overflow \n");
  return;
 }
```

```
    top = top + 1;
    s[top] = item;
}

/* PoP */
void pop()
{
    /* Check if stack is empty */
    if (top == -1)
    {
        printf("Stack Underflow\n");
        return;
    }
    item = s[top];
    top = top - 1;
}
```

ii) STACK:

0	1	2	3	4	5	6	7
A	C	D	F	K			

a)

0	1	2	3	4	5	6	7
A	C	D	F				

ITEM = K.

TOP = 3

b)

0	1	2	3	4	5	6	7
A	C	D					

ITEM = F

TOP = 2

c)

0	1	2	3	4	5	6	7
A	C	D	L				

TOP = 3

d)

0	1	2	3	4	5	6	7
A	C	D	L	P			

TOP = 4

e)

0	1	2	3	4	5	6	7
A	C	D	L				

ITEM = P

TOP = 3

f)

0	1	2	3	4	5	6	7
A	C	D	L	R			

TOP = 4

g)

0	1	2	3	4	5	6	7
A	C	D	L	R	S		

TOP = 5

h)

0	1	2	3	4	5	6	7
A	C	D	L	R			

ITEM = S

TOP = 4.

Q7. Infix to Postfix.

(A + B) * (C ^ (D - E) + F) - G)

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

	Symbol	Stack	P.
		(	
1.	(	((	
2.	A	((	A
3.	+	((+	A
4.	B	((+	A B
5.	)	(	A B +
6.	*	(*	A B +
7.	(	(*(	A B +
8.	C	(*(	A B + C
9.	^	(*(^	A B + C
10.	(	(*(^(	A B + C
11.	D	(*(^(	A B + C D
12.	-	(*(^(-	A B + C D
13.	E	(*(^(-	A B + C D E
14.	)	(*(^	A B + C D E -
15.	+	(*(+	A B + C D E - ^
16.	F	(*(+	A B + C D E - ^ F

	Symbol	Stack	P
17.	)	(*	AB+CDE-^F+
18.	-	(-	AB+CDE-^F+*
19.	G	(-	AB+CDE-^F+*G
20.	)		AB+CDE-^F+*G-

P: = AB+CDE-^F+*G-

Q.2. Create array of structure to represent the polynomials. Add 2 and store in third, Q.6 Let S and T be 2 strings:-

S= 'I AM PROUD'

T= 'TO BE AN INDIAN'

i) find the length of S and T = 10 and 15 (2)

ii) find the SUBSTRING(S, 4, 5) = 'M PRO'

SUBSTRING(T, 10, 5) = 'INDIA' (2)

iii) find INDEX(S, 'P') = 6

INDEX(T, 'THEN') = 0 (2)

iv) DELETE('AAABBB', 3, 3) = 'AAB'

~~REPLACE(T,~~

INSERT('AAA', 2, 'BBB') = 'ABBBAA' (4)

REPLACE('ABABAB', 'B', 'BAB') = 'ABABABAB'

Give S//T := 'I AM PROUD TO BE AN INDIAN'

Q5/ Solⁿ. to the Towers of Hanoi prob. for n=1 and n=2.

for n=1: A→C   Only 1 move.
n=2: A→B, A→C, B→C.   3 moves

Rather than finding a separate solⁿ. for each n, we use the technique of recursion to develop a general solⁿ.

↳ Solⁿ. to the Towers of Hanoi prob. for n>1 disks may be reduced to the following subprobs.

(1) Move the top (n-1) disks from peg A to peg B.

(2) Move the top disk from peg A to peg C ; A→C

(3) Move the top (n-1) disks from peg B to peg C.

TOWER (N, BEG, AUX, END) procedure which moves the top N disks from the initial peg BEG to the final peg END using the peg AUX as an auxiliary.

∴ when N=1, we have

TOWER (1, BEG, AUX, END) consists of the single instruction BEG→END

When n>1, the solⁿ. may be reduced to the solⁿ. of the follow. 3 subprobs:— (i.e. Recursive Solution)

(1) TOWER (N-1, BEG, END, AUX)

(2) TOWER (1, BEG, AUX, END) or BEG→END.

(3) TOWER (N-1, AUX, BEG, END).

Algo

TOWER(N, BEG, AUX, END)

This procedure gives a recursive sol^n. to the Tower of Hanoi problem for N disks.

1. If N=1, then:
   (a) Write: BEG→END.
   (b) Return.
   [End of If structure].

2. [Move N-1 disks from peg BEG to peg AUX.]
   Call TOWER(N-1, BEG, END, AUX).

3. Write: BEG→END

4. [Move N-1 disks from peg AUX to peg END.]
   Call TOWER(N-1, AUX, BEG, END).

5. Return.

Thus, we can view this sol^n. as a divide and conquer algorithm, since the sol^n. for n disks is reduced to a sol^n. for (n-1) disks and a sol^n. for n=1 disks