DSA MIDSEM QUESTION 2020

1. Answer the following questions.
   a. What is ADT? Implement push() operation using array data structure.
   b. What is ADT? Implement pop() operation using array data structure.
   c. What is ADT? Implement push() operation using linked list data structure.
   d. What is ADT? Implement pop() operation using linked list data structure.

2.
   a. A matrix B[10][20] is stored in the memory with each element requiring 4 bytes of storage. If the address of B[2][1] is 2140, write the formula to solve this problem and calculate the address of B[5][4] when the matrix is stored in Column Major Order.
   b. A matrix B[10][20] is stored in the memory with each element requiring 2 bytes of storage. If the base address at B[1][2] is 2140, write the formula to solve this problem and calculate the address of B[5][4] when the matrix is stored in Row Major Order.
   c. A matrix B[20][10] is stored in the memory with each element requiring 2 bytes of storage. If the base address at B[1][2] is 2140, write the formula to solve this problem and calculate the address of B[5][4] when the matrix is stored in Column Major Order.
   d. A matrix B[20][10] is stored in the memory with each element requiring 4 bytes of storage. If the base address at B[2][1] is 2140, write the formula to solve this problem and calculate the address of B[5][4] when the matrix is stored in Row Major Wise.

3.

a. Given is one head pointer pointing to the head node of circular double linked list. Write a C function/pseudo-code to delete the last node without traversing the linked list.

b. Given is one head pointer pointing to the head node of circular single linked list. Write a C function/pseudo-code to delete the immediate before node of the last node (last but one node).

c. Given is one pointer ptr is pointing to any intermediate node of circular double linked list. Write a C function/pseudo-code to delete the previous node of ptr without traversing and without taking any extra pointer.

d. Given is one pointer ptr is pointing to any intermediate node of circular double linked list. Write a C function/pseudo-code to delete the next node of ptr without traversing and without taking any extra pointer.

4.
   a. Following is the infix expression. Let the infix to postfix conversion process is applied on this. Write the stack[] content when number of elements in the stack will be of maximum. Infix[]={A+(B*(D/E^F)-D)+P}.

   b. Following is the infix expression. Let the infix to postfix conversion process is applied on this. Write the corresponding postfix[] content when number of elements in the stack will be of maximum. Infix[]= {A^(B/A*(C+D)-E)+F}

c. Following is the infix expression. Let the infix to prefix conversion process is applied on this. Write the stack[] content when number of elements in the stack will be of maximum. Infix[]={A+(B*(D/E^F)-D)+P}.

d. Following is the infix expression. Let the infix to prefix conversion process is applied on this. Write the corresponding prefix[] content when number of elements in the stack will be of maximum. Infix[]={ A^(B/A*(C+D)-E)+F}

5.

a.
```
int i, m, l = 0;
for (i = m / 2; i <= m; i++) {
      for (k = 2; k <= n; k = k * 2) {
            l = l + m / 2;
      }
}
```

i. O(n^2)
ii. O(nlogm)
iii. O(mlogn)
iv. O(logn^2)

Ans: iii

b.
```
int abc(int n)
{
   int I, j, count = 0;
   for (int j = 0; j < n; j++)
       xyz(j);
   return count;
}
xyz(int p)
{
   for (int i = p; i > 0; i--)
            count = count + 1;
}
```

i. O(n^2)
ii. O(nlogn)
iii. O(n)
iv. O(log^n)

Ans: i

c.
```
void abc(int n, int a[])
{
      int k = 0, l = 0;
      for(; k < n; ++k)
            while(l < n && a[k] < a[l])
                  l++;
}
```

i. O(n)
ii. O(n^2)

iii. O(nlogn)
iv. O(n(logn)^2)

Ans: i

d. Let the following function reverses the length n number num=$K_1K_2K_3..K_n$ [K_i=value at ith position of the number]

```
int num, reverse;
//Take input for num
reverse = 0;
while (num > 0)
{
    rev = rev*10 + num%10;
    num = num/10;
}
```

i. num = $K_1K_2K_3..K_{n-i}$ and reverse = $K_mK_{m-1}..K_{m-i+1}$
ii. num = $K_1K_2K_3..K_{n-i}$ and reverse = $K_mK_{m-1}..K_{m-i-1}$
iii. num = $K_mK_{m-1}..K_{m-i-1}$ and reverse = $K_1K_2K_3..K_{n-i}$
iv. None

Ans: i

Section B

1. Let the individual school of the university is maintaining the BTech student information using the linked list data structure with student information as name (string), roll-number (int), semester(int), section(char), and CGPA (float). Write the structure to define of the node of the linked list.
   Suppose for the class/semester promotion the university wants to apply the following changes based on student's CGPA. Let after every semester result the student database is updated as follows.

a. For the semester 1st to 7th let following changes have to make

For CGPA	Semester	Section
CGPA < 6	Delete the node from the master linked list and add to another linked list.	
6<=CGPA<7	Semester promotion	Section will change to 'A'
7<=CGPA<8	Semester promotion	Section will change to 'B'
8<=CGPA<9	Semester promotion	Section will change to 'C'
9<=CGPA<10	Semester promotion	Section will change to 'D'

b. For the semester 8th delete the student information or delete the node.

Write a program to implement this.

2. Let given is one stack where the elements are stored with their number of occurrences. Using the basic stack push() and pop() operations implement the following insert() and delete() functions.
   a. Insert():- insert one new element in the stack using the push() or/and pop() operation(s) such that if the element exist then it will just increase the

number of occurrences or if element doesn’t exist then it will insert the element with its occurrence as 1.

b. Delete():- delete the existing element from the stack using the push() or/and pop() operation(s) such that if the element exist then it will just decrease the number of occurrences or if element doesn’t exist then it will give the underflow message.

Example: Let the stack contains the following elements.

Element	Occurrences
15	2
9	1
4	5
17	3
5	1

Insert(4):-

Element	Occurrences
15	2
9	1
4	5+1=6
17	3
5	1

Insert(3):-

Element	Occurrences
3	1
15	2
9	1
4	5
17	3
5	1

Insert(4):-

Element	Occurrences
15	2
9	1
4	5-1=4
17	3
5	1

Delete(5):-

Element	Occurrences
15	2
9	1
4	5-1=4
17	3

3.

a. What is the requirement to convert any infix expression to it’s corresponding prefix or postfix notation? What is the requirement of putting one closing bracket at the end of the infix[] before converting to postfix expression? Using the conversion algorithm, stepwise translate the following infix expression to its corresponding postfix expression. And then do the evaluation of the postfix expression using the stack data structure.

Infix[]= (A-(B+C)/F)^C+G/E where A=10, B=5, C=3, F=2, G=16, E=4

b. Write an algorithm/ C-function to identify the position of an unmatched parenthesis in a given mathematical expression using stack. For Example: (x+y)*((z-w), Output: if the expression starts from 0 position, in 6th position the left parenthesis '(' is an unmatched parenthesis.