Sardar Patel Institute of Technology
Bhavan's Campus, Munshi Nagar, Andheri (West), Mumbai-400058, India
(Autonomous College Affiliated to University of Mumbai)

Mid Semester Examination

March 2020

Max. Marks: 20	Duration: 1 Hr
Class: F.Y.M.C.A	Semester:I
Course Code: MCA23	Branch:M.C.A.
Date: 4/3/20	
Time: 12 to 1	
Name of the Course: Data Structures	

Instruction:

(1) All questions are compulsory

(2) Draw neat diagrams

(3) Assume suitable data if necessary

Q.1 Compare worst case complexity of Insertion sort and Selection sort by considering following data and identify which one is best.
Arr[6] = {10, 9, 8, 5, 4, -2}

Explanation : Arr[6] = {10, 9, 8, 5, 4, -2}

Insertion sort : **(2 mks)**

Number of swapping and comparison in first pass = 4
Number of swapping and comparison in second pass = 8
Number of swapping and comparison in third pass = 12
Number of swapping and comparison in forth pass = 16

Hence the complexity = 4 + 8 + 12 + 16 = O(n^2)

Selection Sort : **(2 mks)**

Number of swapping and comparison in first pass = 8
Number of swapping and comparison in second pass = 7
Number of swapping and comparison in third pass = 6
Number of swapping and comparison in forth pass = 5

Hence the complexity = O(n^2) using A.P.

Identification of which one is best (Insertion sort) **(1 mk)**

Q.2 Apply Fold Boundary technique with key offset method for mapping following data in memory size 23.
12345, 81, 435563, 5435, 56761

Explanation : Mapping each key to its proper location. **(1 mk each)**

1) H(12345) = 10 + 23 + 54 = 87 % 23 = **18**
2) H(81) = 18 Collision Hence offset = key / no. Of locations = 81 / 23 = 3.5 = 4
Hence New address = Old address + Offset = 18 + 4 = **22**
3) H(435563) = 34 + 55 + 36 = 125 Ignore carry. Hence 25 % 23 = **2**
4) H(5435) = 45 + 53 = 98 % 23 = 6
5) H(56761) = 50 + 67 + 16 = 133 ignore carry hence 33 % 23 = **10**

Q.3 Apply push and pop operations to evaluate following Postfix expression using Stack of size 5. Construct an algorithm for the same.

P = 12 34 + 30 - 20 25 + *

Explanation: Table Creation **(2 mks)**

Symbol	Stack
12	12
34	12 34
+	46
30	46 30
-	16
20	16 20
25	16 20 25
+	16 45
*	720
)	

Algortihm: **(3 mks)**

1) Add a right parenthesis ")" at the end of the expression P.
2) Scan P from left to right and repeat step 3 and 4 for each symbol of P until ")" is encountered.
3) If an Operand is encountered, push it in top of the stack.
4) If an Operator OP is encountered ,
   A) Pop top two elements from stack, when A is the top element and B is second top element.
   B) Evaluate B OP A
   C) Push the result of B) back on top of the stack.
5) Set value equal to the top element of the stack.

OR

Q. 3 Apply Enqueue (E) and Dequeue (D) operations to store following data in circular queue of size 4 and show final content of the circular queue. Also construct algortihm for Enqueue operation for circular queue.

E1, E2, E3, D, D, E4, E5, D, D, E6

***Explanation** : Enqueuing and Dequeuing all elements to its proper place **(0.25 mk each)**
Showing final content **(0.5 mk)**

Index	0	1	2	3
Data	5	6		

Enqueue operation Algorithm **(2 mks)**

Initialze the front and rear to -1 , queue is an array of size 4. MAXZISE is the maximum size of a queue here in this case it is 4.

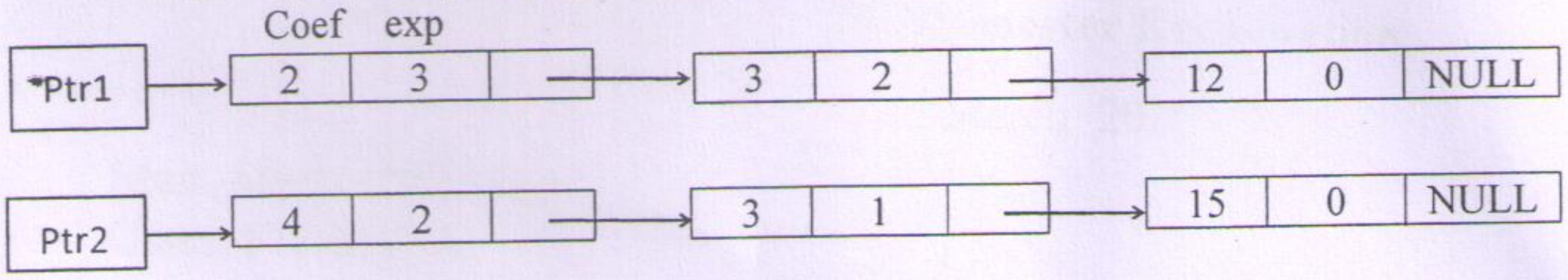
1) Check for Overflow condition
2) If overflow is not there, perform step 3 to 5.
3) If front is equal to -1, reset front to 0.
4) rear = (rear + 1) % MAXSIZE
5) queue[rear]=element

Q.4 Select appropriate linked list to add following polynomial equations and Construct an algorithm and conclude your answer.

$$2x^3 + 3x^2 + 12$$
$$4x^2 + 3x + 15$$

Explanation :

Identification of appropriate linked list (Singly linked list) **(1 mk)**



Algorithm **(4 mks)**

```
Polynomial_Addition(Poly1,Poly2,Poly)
{
        Ptr1= Poly1, Ptr2 = Poly2, Ptr = new node(), Poly = Ptr;
        WhilePtr1!=NULL && Ptr2!=NULL)
        {
                If(Ptr1->exp > Ptr2->exp)
                {
                                Ptr->coef = Ptr1 ->coef;
                                Ptr->exp = Ptr1->exp;
                                Ptr1 = Ptr1->next;
                }
                Else if (Ptr1->exp < Ptr2->exp)
                {
                                Ptr->coef = Ptr2 ->coef;
                                Ptr->exp = Ptr2->exp;
                                Ptr2 = Ptr2->next;
                }
                Else
                {
                        Ptr->coef = Ptr1->coef + Ptr2.coef;
                        Ptr->exp = Ptr1->exp;
                        Ptr1 = Ptr1->next;
                        Ptr2 = Ptr2->next;
                }

                Ptr->next = new node();
                Ptr = Ptr->next;
        }
```