Roll No.:........................

National Institute of Technology, Delhi

Name of the Examination: B.Tech. / M.Tech. / Ph.D.

Mid-Semester Examination Spring, 2020

Branch	**: CSE+ECE+EE (BTech)**	**Semester**	**: 2nd**
Title of the Course	**: Data Structure**	**Course Code**	**: CSB102**

Time: 1.5 Hours **Maximum Marks: 25**

Note : All Questions are compulsary. The Questions in each section should be answered at one place.

Section A (8 marks)

Q1. Determine the complexity in terms of big O for the following using step count method: **(1)**

a.
```
function(int n) {
   for (int i = 0; i<n; i++)
      for(int j=i; j<i*i; j++)
         if (j %i == 0){
            for (int k = 0; k < j; k++)
               printf(" * ");
         }

}
```

b.
```
function (int n) {
for  ( i=1; i<n;  i =2*i )
{
k= n;
while (k > 0)
k=k / 2;
} }
```

Q2. Determine the values of constants c and n0 for the following **(1+1.5)**

a. $10\ n^3 + 15\ n^4 + 100\ n^2 .\ 2^n\ = O\ (\ n^2 . 2^n\ )$

b. $\sum_0^n i^2\ =\ \theta\ (n^3\ )$

Q3. Consider the following expression **4 / 2 – 3 + 6 * 5 - 4 * 3** **(3)**

a. Convert the expression into postfix using stacks

b. Convert it to prefix expression using stacks

c. Evaluate the postfix expression obtained in **part "a"** using stacks.

Q4. Write a C function **IsPrime (int n),** which will check whether "n" is a prime number or not. [Hint use sqrt function]. Also determine its best and worst case time complexity. **(1.5)**

Section B (8 marks)

Q5. Explain an efficient way of storing a sparse matrix in the memory. **(2)**

Q6. There are n people in a room. If each person shakes hands once with every other person. What is the total number of handshakes? Represent your problem in form of recursion. Which data structure will be best suited in this scenario, Justify your answer. **(3)**

Q7. Consider the stack data structure, supporting two operations push and pop. Suppose that for the below 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. **(3)**

Consider the following sequence of letters and asterisks:

EAS*Y*QUE*ST***IO*N*****

a) Show the sequence of values returned by the pop operations.

b) Determine the number of times the PUSH and POP operations are performed.

Section C (9 marks)

Q8. Two dimensional integer array TABLE [4] [6] is stored in row major order with base address 100. What is the address of the data stored at the location of TABLE [2] [4] ? (Note : compiler consider integer to be of 4 bytes, please specify assumption if any) **(2)**

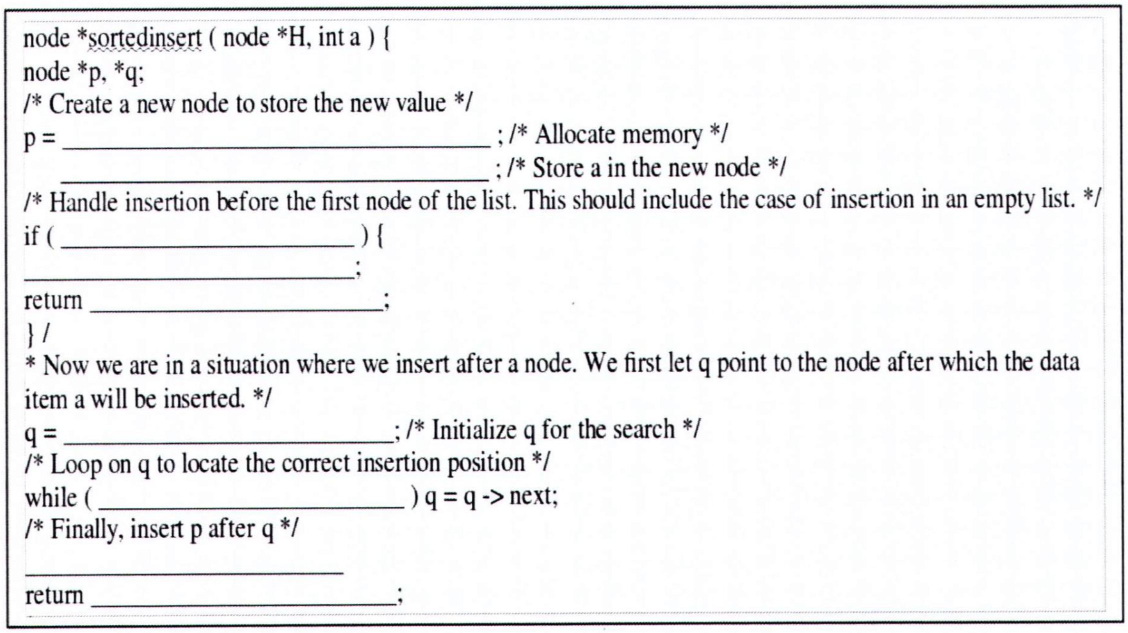
Q9. Discuss the case of overflow and underflow condition in stack using a pseudo code with array and linked list respectively. **(3)**

Q10. In the following, we deal with linked lists which are kept sorted in the increasing order of the data values. To this end, we define a node in the list in the usual way as:

```
typedef struct _node {
int data;
struct _node *next;
} node;
```

A list is specified by a pointer to the first node in the list. We assume that all the nodes in the list store valid data items, that is, there is no dummy node at the beginning of the list.

Let **H** be a pointer to the first node in a sorted linked list. **H** is **NULL** if the list is empty. We plan to insert a data value **a** in the list such that the list continues to remain sorted after the insertion. The value **a,** to be inserted, may or may not be already present in the list. Complete the following function to carry out this sorted insertion. The function should return a pointer to the first node in the modified list. **(4)**



```
node *sortedinsert ( node *H, int a ) {
node *p, *q;
/* Create a new node to store the new value */
p = ________________________________ ; /* Allocate memory */
    ________________________________ ; /* Store a in the new node */
/* Handle insertion before the first node of the list. This should include the case of insertion in an empty list. */
if ( ______________________ ) {
________________________;
return ______________________;
} /
* Now we are in a situation where we insert after a node. We first let q point to the node after which the data
item a will be inserted. */
q = ______________________; /* Initialize q for the search */
/* Loop on q to locate the correct insertion position */
while ( ______________________ ) q = q -> next;
/* Finally, insert p after q */
________________________
return ______________________;
```