Sub Name Code: DSA
Subject Code: CS-2001
Program Name: B.Tech
Semester: III (Regular)
Year - 2019

AUTUMN MID-SEMESTER - 2019
KIIT, Deemed to be University, Bhubaneswar-24
Data Structure and Algorithm
CS-2001

Time: $1^{1/2}$ Hours **Full Mark: 20**

The figures in the margin indicate full marks. Candidates are required to give their answers in their own words as far as practicable and all parts of a question should be answered at one place only.

(Answer any four questions including question No.1 which is compulsory)

Q1.	Answer all the following questions. Provide appropriate example, if necessary. [1×5 =5]

(a)	Find the time complexity of the following code:
	
	for (i=1;i<=m;i=i×2)
	for (j=1;j<=i;j++)
	printf("%d %d ",i,j);
	
	Evaluation Scheme:
	1 mark for the correct answer.
	
	Solution:
	For i=1 => j executes 1 time
	For i=2 => j executes 2 times
	For i=4 => j executes 4 times
	•
	•
	For i=m => j executes m times
	Total number of execution = 1+2+4+8+...+m
	$T(n) = 2^0+2^1+2^3+...+2^{logm}$
	$= 2^{logm+1}-1$
	$= O(2^{logm}) = O(m)$ or $O(n)$

(b)	In a two dimensional matrix A[0,..,r-1, 0..,c-1] the location of A[i, j] is same in both row-major-order and column-major-order. Find the relationship between the number of rows and number of columns (r, c) with the cell index (i, j) and justify the relation with one example. **Evaluation Scheme:** 1 mark for the correct answer. Formula of both order is of 0.5 mark and finding relation is of 0.5 mark. **Solution:**

	Row Major Order: Loc(A[i, j]) = Base + ((i-0) ×c + (j-0)) ×w Col Major Order: Loc(A[i, j]) = Base + ((i-0) + (j-0) ×r) ×w Equating both: Base + ((i-0) ×c + (j-0)) ×w = Base + ((i-0) + (j-0) ×r) ×w => i×c + j = i + j×r => i×(c-1) = j×(r-1) => $\frac{i}{j} = \frac{c-1}{r-1}$
(c)	Using malloc() write the code section to do the memory allocation for one matrix of size M×N. **Evaluation Scheme:** 1 mark for the correct answer. **Solution:** int **a a = (int* *) malloc (m × sizeof(int *)); for (int i; i<m; i++) a[i] = (int *) malloc (n × (sizeof(int));

(d)	Write algorithm or pseudo code for deleting all the nodes from the single linked list which ar divisible by a given number k.
	
	Evaluation Scheme:
	1 mark for the correct answer. Step mark may be awarded by looking into the partial correctness of the answer.
	
	Solution:
	void delete()
	{
	struct node *ptr = start, *prev;
	while(ptr != NULL)
	{
	if (ptrinfo%k==0)
	{
	if(start == ptr)
	{
	start = start → next;
	free(ptr);
	ptr = start;
	}
	else
	{
	prev →next = ptr;
	free(ptr);
	ptr = ptr→ next;
	}
	}
	else

	{
	prev=ptr;
	ptr= ptr→next;
	}
	}

(e)	Write a recursive function to display elements of stack using the operation push() and pop() and without violating the LIFO concept.
	
	Evaluation Scheme:
	1 mark for the correct answer.
	
	Solution:
	void display()
	{
	int x;
	if(top = = -1)
	return;
	else
	{
	x=pop()
	display();
	printf("%d", x);
	push();
	}
	}

Q2.	[2.5+2.5]
(a)	Given two sorted arrays A and B. Array A is full and array B is partially full and the number of empty slots are just enough to accommodate all elements of array A. Write one C function to merge the two sorted arrays to fill the array B without considering the third array. **Evaluation Scheme:** 2.5 mark for the correct answer. 1.5 mark for shifting all empty space to one side and 1 mark for bringing the array A[] element to array B[]. **Solution:** Step1: Shift all the blank space or empty space to right side of the B[] Step2: Do the merging in B[] to keep all the element in B[] either direct bringing from A[] or by doing the proper merging process. Note- Full mark also can be awarded even for the answer which is not getting the sorted

output array.

```
void merge(int A[], int B[], int m, int n)// size of A[] is m and size of B[] is n
{
   int i=0, j=n-1; //Index to be used in array B[]
   // Shift all the blank space to the right side
   while(i < j)
   {
     while(B[i]!=' ')
          i++;
     while(B[j] = = ' ')
          j--;
     if(i<j)
     {
       B[i] = B[j];
       B[j] = ' ';
      }
    }
    size=i
    // Either do the merging as follows to have proper sorted element
    i=0, j=0;
    while(i<m && j<n)
   {
     if(A[i] < B[j])
     {
       for(k=j; k<n-1;  k++)
            B[k+1] = B[k];
      B[j] = A[i];
      i++;   j++;
    }
  else
     j++;
 }
  // Or just copy the element of A[] to B[]
   for (i=0;j=size; j<m; i++, j++)
        B[j] = A[i];
```

(b) Let the number "XYZ" is represented as Z->Y->X in the linked list. If two such numbers are given in two different linked list then with the above representation write a C-function to subtract the second number (represented in second linked list) from the first number (represented in first linked list) and store the result in third linked list with above representation.

Note: In a single traversal the code should do the subtraction.

Example: Linked list1:- 1→0→0→2 (List for number 2001)
Linked list2:- 8→7→5 (List for number 578)
Resultant List:- 3→2→4→1 (List for number 1423 i.e 2001-578=1423)

Evaluation Scheme:

2.5 mark for the correct answer. Step mark may be awarded by looking into the partial correctness of the answer.

Solution:

```
void Subtract(Struct node *start1, Struct node *start1)
{
  struct node *start3 = NULL, *p = start1, *q = start2, *k, *r;
  while(p != NULL && q != NULL)       {
    if(p →info ≥ q→ info) // possible to direct subtract the corresponding element
    {
      add_end((&start3, p→info - q→info)// add new node in start3 at the end
      p = p →next; q= q → next;
    }
    else if(p → next→info >0)// possible to borrow from just next element
    {
      p → info = p →info + 10;
      p → next → info =  p → next → info -1;
    }
    else
    {
      k = p → next;
      while (k → info = = 0) // search the element which can give the borrow
          k= k → next;
      p → info = p → info + 10;
      for( r= p → next; r != k; r = r → next)
           r→ info =9;
      k → info = k → info -1;// the element which has given the borrow
    }
    }
}
```

Q3.		[2+3]
(a)	Write a C function which will take one unsorted linked list with redundant elements and will output one sorted linked list with unique elements. **Evaluation Scheme:** 2 mark for the correct answer. 1 Mark for the duplicate remove and 1 mark to do the sorting.Step mark may be awarded by looking into the partial correctness of the answer. **Solution:**	

```
void Rem_dupl&sort()
{
  struct node *p1, *p2, *prev;
  int t;
  for(p1= start; p1 != NULL; p1 = p1→next)// duplicate remove
  {   prev = p1;
      for(p2 = p1→ next; p2 != NULL;)
          if(p1 → info = = p2→ info)
          {
            prev → next = p2 → next;
            free(p2);
            p2 = prev → next;
```

```
        }
        else
        {
          prev=p2;
          p2 = p2 → next;
        }
      }
      for(p1= start; p1! = NULL; p1 = p1→next)// Sorting
        for(p2= start; p2! = NULL; p2 = p2→next)
            if(p1 → info > p2 → info)
            {
              t = p1 → info;
              p1 → info = p2 → info;
              p2 → info = t;
            }
```

(b) What are the different way of representing sparse matrix in memory. Explain through example. Using any suitable representation write the C-function to multiply two sparse matrices.

Evaluation Scheme:

3 mark for the correct answer. 1 Mark for the Sparse matrix representation with one example and 2 mark to do the multiplication. Step mark may be awarded by looking into the partial correctness of the answer.

Solution:

There are two way of representing the sparse matrix, array triplet and linked list triplet. Small example in each case.

```
struct node                    struct head
{ int r, c, ele;               { int tr, tc, tele;
 struct node *next;};            struct node *next;};

void multiply_sparse(struct head *head1, struct head *head2, struct head **head3)
{
 if(head1 → tc != head2 → tr)   return;
 *head3  → tr = head1  → tr;
 *head3  → tc = head2  → tc;
  struct node *p1, *p2, *prev;
  int c=0;

  for(p1= head1→next; p1! = NULL; p1 = p1→next)
        for(p2= head2→next; p2! = NULL; p2 = p2→next)
            if(p1 → c = = p2 → r)
            {
                add_end(head3, p1 → r, p2 → c, p1 →info × p2 → info) // this
                    function insert the new node at the end of 3rd sparse matrix head3
                c++; // Initially count the number of node in 3rd sparse matrix
```

```
                }
      for(p1= head3→next; p1!= NULL; p1 = p1→next)// duplicate remove
       {   prev = p1;
          for(p2 = p1→ next; p2 != NULL;)
             if(p1 → r = = p2→ r && p1 → c = = p2→ c)
             {
               p1 → info = p1 → info + p2 → info
                prev → next = p2 → next;
                free(p2);
                p2 = prev → next;
                c--; //Number of nodes decreased due to duplicate removal
             }
             else
            {
               prev=p2;
               p2 = p2 → next;
            }
        }
        *head3 → tele = c;
   }
```

Q4. [2.5+2.5]

(a) Write a C-function to reverse the nodes of a double linked list and justify its correctness with suitable example.

Evaluation Scheme:

2.5 mark for the correct answer. 2 mark for the reversing process and 0.5 for the example.

Solution:

```
    void reverse()
   {
     struct node *p1, *p2;
     p1 = start;
     p2 = p1 → next

     p1 → next = NULL;
     p1 → prev = p2;
    while(p2 != NULL)
    {
      p2→prev = p2→next
      p2→next = p1;

      p1=p2;
      p2 = p2→next
     }
     start = p1;
  }
```

Illustrate with one small example.

(b) Compare the representation of polynomials using array and linked list. Write a C-function to add two polynomials (polynomial with two variables) represented using linked list.

e.g. $P(x , y) = 7x^5y^3 + 8x^4y^3 - 3y^3$ .....

Evaluation Scheme:

2.5 mark for the correct answer. Step mark may be awarded by looking into the partial correctness of the answer.

Solution:

```
 struct node

 {   int coff, exp1, exp2;};

void add(struct head *poly1, struct head *poly2)
{
  struct node *p1, *p2, *prev, *q;
  if (poly1= = NULL)//join two linked list
      poly1=poly2;
  else
  {
     q = poly1;
     while(q→next != NULL)
          q = q→next;
     q→next = poly2;
  }
    for(p1= poly1; p1!= NULL; p1 = p1→next)// duplicate remove
     {   prev = p1;
        for(p2 = p1→ next; p2 != NULL;)
           if(p1 → expo1 = = p2→ expo1 && p1 → expo2 = = p2→ expo2)
           {
             p1 → coff = p1 → coff + p2 → coff
              prev → next = p2 → next;
              free(p2);
              p2 = prev → next;
            }
            else
            {
             prev=p2;
             p2 = p2 → next;
             } } }
```

Q5.	[2+3]
(a)	What are the different data structure used to represent the stack ADT. Write the Push() operation using one data structure and Pop() operation using the other data structure on stack ADT.

Evaluation Scheme:
2 mark for the correct answer. 1 Mark for Push() and 1 mark for pop().

Solution:
There are two way of representing stack ADT in memory, one is using array and another one using linked list.
Push() function using array and pop() function using linkedlist
or
Push() function using linkedlist and pop() function using array.

(b) Convert the infix expression, ((a/(b-c+d))*(e-a)*c) to postfix expression. Write the pseudo code to evaluate the postfix expression and trace the result for the evaluation of resultant postfix expression with the given data as a=6, b=3, c=1, d=2, e=4.

Evaluation Scheme:
3 mark for the correct answer. 1 Mark for Conversion, 1 mark for algorithm of evaluation and 1 mark for evaluation.

Solution:
a. Conversion of infix to postfix is either using stack ADT or manually.
abc-d+/ea-*c*
b. Pseudo code of infix to postfix conversion
c. Evaluation: 6,3,1,-,2,+,/4, 6, -, *, 1, *

Symbol	Stack	Rough
6	6	
3	6, 3	
1	6, 3, 1	
-	6, 2	[3 - 1 = 2]
2	6, 2, 2	
+	6, 4	[2 - 2 = 4]
/	1.5	[6 / 4 = 1.5]
4	1.5, 4	
6	1.5, 4, 6	
-	1.5, -2	[4 - 6 = -2]
*	-3	[1.5 * -2 = -3]
1	-3, 1	
*	-3	[-3 * 1 = -3]

OR

Symbol	Stack	Rough
6	6	
3	6, 3	
1	6, 3, 1	
-	6, 2	[3 - 1 = 2]
2	6, 2, 2	
+	6, 4	[2 - 2 = 4]
/	1	[6 / 4 = 1.5]
4	1, 4	
6	1, 4, 6	
-	1, -2	[4 - 6 = -2]
*	-2	[1 * -2 = -2]
1	-2, 1	
*	-2	[-2 * 1 = -2]