AUTUMN MID SEMESTER EXAMINATION-2018

School of Computer Engineering

Kalinga Institute of Industrial Technology
Deemed to be University, Bhubaneswar-24

Data Structure and Algorithm
CS-2001

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

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

Q.1. [5X1]

(a) Write a function code whose upper bound time complexity is O(m^2 + n log m)

Evaluation Scheme – Full mark for the correct answer and no partial marks to be awarded.
Solution –

```
void fun(int m, int n)
{
  int i,j;
  int c =2; // c can be any number
  for (i = 1;i<=m; ++i) {
    for (j = 1;j<=m; ++j)
    printf("[%d,%d]", i,j); //O(1) statement
  }
  for (i = 1;i<=n; ++i) {
    for (j = 1;i<=m; i=i*c)
    printf("[%d,%d], i,j); //O(1) statement
 }
} //end of function
```

O(m^2) (first block)

O(n log m) (second block)

(b) An array, arr[-15……….10, 15……………40] requires four bytes of storage for each element. If the base address of arr is 2500, determine the address of arr[8][20] for row-major and column-major order.

Evaluation Scheme – 0.5 marks for row-major order and 0.5 marks for column-major order. No partial marks to be awarded.
Solution –
[**Note**: Refer to class note for the below formula details]
The address of a location for Row Major order: Address of A[i][j] = BA + w * [n * (i – Lr) + (j – Lc)]
The address of a location for Column Major order: Address of A[i][j] = BA + w * [(i – Lr) + m * (j – Lc)]

Number or rows m = (Ur – Lr) + 1 = [10 – (- 15)] +1 = 26
Number or columns n = (Uc – Lc) + 1 = [40 – 15)] +1 = 26

Row Major order: arr[8][20] = 2500 + 4* [26 * (8 – (-15))) + (20 – 15)] = 4912
Column Major order: arr[8][20] = 2500 + 4 * [(8 – (-15)) + 26 * (20 – 15)] = 3112

(c) Write a recursive function code to count the length of a given grounded header linked list.

Evaluation Scheme – Full mark for the correct answer and partial marks to be awarded depending on the correctness of the steps.

Solution 1 – Considering length, inclusive of the header node

```
int length(struct node *head)
{
   If (head == NULL) //if header is NULL
     return 0;
   return 1 + length(head->next);
}
```

Solution 2 – Considering length, exclusive of the header node

```
int length(struct node *head)
{
   If (head == NULL or head->next == NULL) //if header is NULL or header node is NULL
     return 0;
   return 1 + length(head->next);
}
```

(d) Write a function code to delete the middle most item from the stack wherein push, pop and isEmpty and peek operations are given.

Example:

Input: Stack = [11, 22, 33, 44, 55]
Output: Stack = [11, 22, 44, 55]

Input: Stack = [10, 20, 30, 40, 50, 60]
Output: Stack = [10, 20, 40, 50, 60]

Evaluation Scheme – Full mark for the correct answer and partial marks to be awarded depending on the correctness of the steps.

Solution –

Let's assume s1 is the stack containing n items and s2 is the empty stack. The approach is to pop floor(n/2) {for odd} or floor(n/2) – 1 {for even} elements from s1 and push to s2. Then, delete the topmost element from s1 and push back all elements of s2 to s1.

Pseudocode:

```
int k = floor(n/2);

If (n%2 == 0)
  k = k -1;

for (int i = 1; i<= k; i++)
 push(s2, pop(s1);

pop(s1); //deleting the middlemost element

while (!isEmpty(s2))
  push(s1, pop(s2));
```

(e) Explain head and tail recursion with suitable examples.

Evaluation Scheme – weightage for the definition and example of head recursion is 0.5 marks and of tail recursion 0.5 marks. Partial marks to be awarded depending on the correctness of the steps.
Solution –

<table>
<tr>
<td>Head recursion - the recursive step comes at the beginning of the function, possibly after/within the base criteria.

Function code example -<pre>
FUNCTION count(number)
    IF (number > 0) THEN
        RETURN count(number - 1)
    END IF
    RETURN 0
END FUNCTION</pre></td>
<td>Tail recursion - the recursive step comes last in the function.

Function code example –<pre>
FUNCTION count (number)
    IF(number == 0) THEN
        RETURN 0
    END IF
 RETURN count(number - 1)
END FUNCTION</pre></td>
</tr>
</table>

Q.2. [5]

Two arrays are given and write the function code/ pseudo code / algorithm to swap elements from each array such that after swapping sum of the both arrays is same.

Example:

Array 1: 3, 2, 10, 12 Input: Stack = [10, 20, 30, 40, 50, 60]
Array 2: 6, 4, 9, 10 Output: Stack = [10, 20, 40, 50, 60]

After swapping 3 (from Array 1) with 4 (from Array 2)

Array1: 4, 2, 10, 12 Sum = 28
Array2: 6, 3, 9, 10 Sum = 28

Evaluation Scheme – Full mark for the correct answer and partial marks to be awarded depending on the correctness of the steps. **Note** – marks to be judiciously awarded considering the swapping of one or more than one elements from each array.
Solution –
Note: below solution is for swapping of one element from each array.
Suppose we need to swap x and y element of arrays that have sum as sum1 and sum2.
So after swapping sum of both arrays should be same i.e. sum1 - x + y = sum2 - y + x. so x-y should be equal to (sum1-sum2)/2. So the essence is to find out x and y.
Algorithm :-

```
STEP 1. sum1 = sum of the elements of a and sum2 = sum of the elements of b
STEP 2. Sort both the arrays a & b.
STEP 3. Start traversing  a and b from left and check
          IF (a[i]-b[j] == (sum1-sum2)/2) THEN
            return a[i] and b[j]
          ELSE IF (a[i]-b[j]  <  (sum1-sum2)/2) THEN
            i= i+1
          ELSE
            j = j+1
          END IF
```

Function Code :-

```
void sort(int array[], int n)
{
  for (int i = 0 ; i < n; i++) {
    for (int j = 0 ; j < n - i - 1; j++) {
      if (array[j] > array[j+1])
      {
        int swap = array[j];
        array[j] = array[j+1];
        array[j+1] = swap;
      }
    }
  }
} //end of function

void findPair(int A[], int n, int B[], int m)
{
   int sum1=0 , sum2 = 0, I = 0, j = 0;

   for(int i = 0;i<n;i++)
    sum1+= A[i];

   for(int i = 0;i<m;i++)
    sum2+= B[i];

   int sum = (sum1-sum2)/2;

    sort(A, n);  sort(B, m);

    if (sum == 0)
        return;

    while (i < n && j < m)
    {
        if (A[i] - B[j] == sum)
        {
            printf("%d from A and %d from B are the pairs", A[i], B[j]);
            return;
        }
        else if (A[i] - B[j]< sum)
            i++;
        else
            j++;
    }
    printf( "No such pair ");
    return;
}
```

```
int main()
{
    int A[] = {  3, 2, 10, 12 };
    int B[] = { 6, 4, 9, 10};
    findPair(A, sizeof(A) / sizeof(A[0]), B, sizeof(B) / sizeof(B[0]));
    return 0;
}
```

Q.3. [2.5]

(a) How to represent a polynomial using linked list? Write the algorithm / function code / pseudo code to add three polynomials.

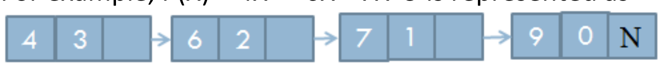
Evaluation Scheme – Representation of the polynomial is 0.5 marks and2 marks for the addition of 3 polynomials. Partial marks to be awarded depending on the correctness of the steps.

Solution – Node of a polynomial is represented in the following way

Coefficient	Exponent	Next

```
struct node
{
 int coefficient;
 int exponent;
 struct node *next;
}
```

For example, P(X) = $4X^3 + 6X^2+7X+9$ is represented as



3 polynomials can be added by calling addition of 2 polynomials twice. For example A, B and C are the 3 polynomials, so add(A, B, C) is equivalent to add(add(A, B), C)

Function code for polynomial addition using linked list –

Let's poly1, poly2 and poly3 are the 3 header pointing to each polynomials.

```
struct node* add(struct node * poly1, struct node * poly2, struct node * poly3)
{
  return polyadd (polyadd(poly1, poly2), poly3);
}

struct node* polyadd(struct node * poly1, struct node * poly2)
{
  struct node * poly = (struct node *)malloc(sizeof(struct Node));
  while(poly1->next && poly2->next)
   {
     // If power of 1st polynomial is greater than 2nd, then store 1st as it is and move its pointer
     if(poly1->pow > poly2->pow)
     {
       poly->pow = poly1->pow;
       poly->coeff = poly1->coeff;
       poly1 = poly1->next;
     }

     // If power of 2nd polynomial is greater than 1st, then store 2nd as it is and move its pointer
     else if(poly1->pow < poly2->pow)
     {
       poly->pow = poly2->pow;
       poly->coeff = poly2->coeff;
       poly2 = poly2->next;
     }
```

```
    // If power of both polynomial numbers is same then add their coefficients
    else
    {
       poly->pow = poly1->pow;
       poly->coeff = poly1->coeff+poly2->coeff;
       poly1 = poly1->next;
       poly2 = poly2->next;
    }

    // Dynamically create new node
    poly->next = (struct Node *)malloc(sizeof(struct Node));
    poly = poly->next;
    poly->next = NULL;
  }

  while(poly1->next || poly2->next)
  {
    if(poly1->next)
    {
       poly->pow = poly1->pow;
       poly->coeff = poly1->coeff;
       poly1 = poly1->next;
    }
    if(poly2->next)
    {
       poly->pow = poly2->pow;
       poly->coeff = poly2->coeff;
       poly2 = poly2->next;
    }
    poly->next = (struct Node *)malloc(sizeof(struct Node));
    poly = poly->next;
    poly->next = NULL;
  }
 return poly;
}
```

(b) Design a suitable data structure to efficiently represent a sparse matrix? Write an algorithm/ function code / pseudo code to add the original sparse matrix with the transpose of the same matrix. [2.5]

Evaluation Scheme – Weightage for the representation of the sparse mark is 0.5 marks, 0.5 mark for the transpose and 1.5 marks for the addition. Partial marks to be awarded depending on the correctness of the steps.

Solution –

2D array (either row or columnar) is used to represent a sparse matrix in which there are three columns named as

Rows: Index of row, where non-zero element is located

Columns: Index of column, where non-zero element is located.

Values: Value of the non zero element located at index – (row, column)

2D array Row Representation:

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 9 & 0 \\ 0 & 8 & 0 & 0 & 0 & 0 \\ 4 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 5 \\ 0 & 0 & 2 & 0 & 0 & 0 \end{bmatrix} \rightarrow$$

Rows	Columns	Values
5	6	6
0	4	9
1	1	8
2	0	4
2	2	2
3	5	5
4	2	2

2D array Columnar Representation:

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 9 & 0 \\ 0 & 8 & 0 & 0 & 0 & 0 \\ 4 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 5 \\ 0 & 0 & 2 & 0 & 0 & 0 \end{bmatrix} \rightarrow$$

Rows	5	0	1	2	2	3	4
Columns	6	4	1	0	2	5	2
Values	6	9	8	4	2	5	2

Addition of the sparse matrix: Below Function code is illustrating 2D array row presentation

```
#define max 10 //just any number
void add(int sparseM[max][3])
{
  If (sparseM[0][0] != sparseM[0][1])
  {
     printf("Not a square sparse matrix, hence transpose is not possible");
     return;
  }

  int transSparseMatrix[max][3]; //declaring a new sparse matrix transpose

  //populating 0th row of the transpose sparse matrix
  transSparseMatrix[0][0] = sparseM[0][1];
  transSparseMatrix[0][1] = sparseM[0][0];
  transSparseMatrix[0][2] = sparseM[0][2];

  int k =1;
  int n = transSparseMatrix[0][2]; //counting the total number of non-zero elements

  //sparse matrix formation
  for (int i=0;i< sparseM [0][1];++i) //traversing through the number of columns
  {
    for (int j=1;j<=n;j++)
    {
      if(i==b1[j][1])
      {
            transSparseMatrix[k][0]=i;
            transSparseMatrix[k][1]= sparseM[j][0];
            transSparseMatrix[k][2]= sparseM[j][2];
```

```
            k++;
        }
      }
    }
//sparse matrix addition
int r,c,i,j,k1,k2,k3,tot1,tot2;

tot1 = sparseM[0][2];
tot2 = transSparseMatrix[0][2];
k1 = k2 = k3 = 1;
int addSparseMatrix[max][3]

while ( k1 <= tot1 && k2 <= tot2)
{
 if ( sparseM[k1][0] < transSparseMatrix[k2][0] )
 {
            addSparseMatrix[k3][0] = sparseM[k1][0];
            addSparseMatrix[k3][1] = sparseM[k1][1];
            addSparseMatrix[k3][2] = sparseM[k1][2];
            k3++;k1++;
 }
 else if ( sparseM[k1][0] > transSparseMatrix[k2][0] )
 {
            addSparseMatrix[k3][0] = transSparseMatrix[k2][0];
            addSparseMatrix[k3][1] = transSparseMatrix[k2][1];
            addSparseMatrix[k3][2] = transSparseMatrix[k2][2];
            k3++;k2++;
 }
 else if ( sparseM[k1][0] == transSparseMatrix[k2][0])
 {
  if ( sparseM[k1][1] < transSparseMatrix[k2][1] )
  {
            addSparseMatrix[k3][0] = sparseM[k1][0];
            addSparseMatrix[k3][1] = sparseM[k1][1];
            addSparseMatrix[k3][2] = sparseM[k1][2];
            k3++;k1++;
  }
  else if ( sparseM[k1][1] > transSparseMatrix[k2][1] )
  {
            addSparseMatrix[k3][0] = transSparseMatrix[k2][0];
            addSparseMatrix[k3][1] = transSparseMatrix[k2][1];
            addSparseMatrix[k3][2] = transSparseMatrix[k2][2];
            k3++;k2++;
  }
  else
  {
            addSparseMatrix[k3][0] = transSparseMatrix[k2][0];
            addSparseMatrix[k3][1] = transSparseMatrix[k2][1];
```

```
            addSparseMatrix[k3][2] = sparseM[k1][2] + transSparseMatrix[k2][2];
            k3++;k2++;k1++;
  }
 } // else
} // while

while ( k1 <=tot1 )
{
           addSparseMatrix[k3][0] = sparseM[k1][0];
           addSparseMatrix[k3][1] = sparseM[k1][1];
            addSparseMatrix[k3][2] = sparseM[k1][2];
            k3++;k1++;
} //while

while ( k2 <= tot2 )
{
            addSparseMatrix[k3][0] = transSparseMatrix[k2][0];
            addSparseMatrix[k3][1] = transSparseMatrix[k2][1];
            addSparseMatrix[k3][2] = transSparseMatrix[k2][2];
            k3++;k2++;
} // while

addSparseMatrix[0][0] = sparseM[0][0];
addSparseMatrix[0][1] = sparseM[0][1];
addSparseMatrix[0][2] = k3-1;
```

Q.4. [3]

(a) Demonstrate on how to implement a stack of integers in C using static array i.e. int s[SIZE] and structure. Write functions: push, pop, peek and isEmpty for this implementation.

Evaluation Scheme – Weightage for the representation of stack using static array & structure is 1 mark. Weightage of each function is 0.5 marks. Partial marks to be awarded depending on the correctness of the steps.

Solution –

```
#define SIZE 25 //size can be any number
struct Stack
{
   int top;
   int s[SIZE];
};

int isEmpty(struct Stack st)
{
   return (st.top == -1);
}

void push(struct Stack st, int x)
{
   if (st.top == SIZE-1)
      return;
   st.s[++st.top] = x;
}

int pop(struct Stack st)
{
   if (isEmpty(st))
      return INT_MIN;
   return st.s[st.top--];
}

int peek(struct Stack st)
{
   return st.s[st.top];
}
```

(b) Evaluate the postfix expression: 8 4 / 5 * 7 3 - + with status of stack after execution of each operation. [2]

Evaluation Scheme – Full mark for the correct answer and partial marks to be awarded depending on the correctness of the steps.

Solution –

8 4 / 5 * 7 3 - +		
Scanned character	Stack state	Operations
8	8	Push(8)
4	8 4	Push(4)
/	2	Pop(), pop(), push(2)
5	2 5	Push(5)
*	10	Pop(), pop(), push(10)
7	10 7	Push(7)
3	10 7 3	Push(3)
-	10 4	Pop(), pop(), push(4)
+	14	Pop(), pop(), push(14)
Output: 14		

Q.5. [5]

Let a single linked list consists of positive integers in such a way that the summation of node values in few continuous nodes matches to the value in the next node. For example, the linked list consists of values 2, 1, 3, 5, 3, 9, 17, 2, 4, 6, where 2+1=3 {i.e. summation of 1st and 2nd node matches to 3rd node}, 5+3+9=17 and so on. Here, no. of summation groups are 2 and values in each summation group are {2, 1}, {5, 3, 9}. Write an algorithm or pseudocode or function code to display the no. of summation groups and the values in each summation group.

Evaluation Scheme – Full mark for the correct answer and partial marks to be awarded depending on the correctness of the steps.

Solution –

```
void compute(struct node *head)
{
   int count=0;
   int sum=0;
   struct node *ptr=head,*cur=head;
   while(ptr!=NULL && ptr->next!=NULL)
  {
      sum += ptr->data;
      if(sum==ptr->next->data)
      {
         printf("SUM=%d\n",sum);
         count++;
         sum=0;
         printf("\nGROUP %d:",count);
         while(ptr->next!=cur)
        {
            printf("%d ",cur->data);
            cur=cur->next;
```

```
        }
        printf("\n");
        cur=ptr=ptr->next->next;
      }
      else
        ptr=ptr->next;
   }
}
```

NOTE: Solution can be written in different ways.

Faculty Consent

Sr#	Name of the Faculty	Sign
1	Dr. Arup Abhinna Acharya	
2	Prof. (Dr.) Samaresh Mishra	
3	Dr. Amulya Ratna Swain	
4	Mr. Gananath Bhuyan	
5	Mrs. Suchismita Das	SDas 6/9/18
6	Mr. Rajat Kumar Behera	Rajat Kumar Behera
7	Prof. (Dr.) Alok Jagadev	
8	Dr. Minakhi Rout	Minakhi
9	Mr. Arup Sarkar	Azarkar
10	Dr. Satarupa Mohanty	SM
11	Dr. Suneeta Mohanty	
12	Mr. Nachiketa Tarasia	
13	Mrs. Mamta Motwani	Mamta
14	Ms. Roshni Pradhan	
15	Ms. Shaswati Patra	S patra
16	Ms. Tanusree Parbat	parbat
17	Dr. Tanmaya Swain	
18	Mr. N. Biraja Isac	N Biraja Isac
19	Ms. Nishita Kindo	Nishita Kindo
20	Dr. Amiya Ranjan Panda	Amiya Ranjan Panda 6/9/18
21	Ms. Lipika Mohanty	Lipika Mohanty
22	Ms. Sasmita Nayak	Sasmita Nayak
23	Mr. Rajdeep Chatterjee	
24	Ms. Santwana Sagnika	

Rajat Kumar Behera
06/09/2018
Course Coordinator