# AUTUMN MID SEMESTER EXAMINATION-2022

School of Computer Engineering
Kalinga Institute of Industrial Technology, Deemed to be University
Data Structure and Algorithms
[CS-2001]

**Time: 1 1/2 Hours** **Full Mark: 20**

***Answer any four Questions including Q.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.***

1. Answer all the questions. [ 1 × 5 ]

a) Find out the time complexity of the following function:

```
void func() {
    int i=1, j;
    while(i<=n) {
      for(j=1; j*j<=n; j++)
          printf("Data Structure…\n");
    }
}
```

Ans:
O(n√n)
Since, while loop will run infinitely, award marks accordingly.

b) Consider a 2-D array arr[3...6, -2...5] requires 4 bytes of memory space for each element of the array. If the array is stored in row-major order having base address 1001, then what will be the address of the array element arr[5, 3]?

Ans:
Cols=5-(-2)=7, BA=1001, i=5, j=3, $l_r$=3, $l_c$=-2, w=4
Addr(arr[5, 3]) = BA + w[Cols(i – $l_r$) + (j – $l_c$)]
= 1001 + 4[7(5-3) + (3-(-2))] =1001 + 4[14+5] = 1077

c) If the node of a doubly linked list is declared in C as:

```
struct node {
    int data;
    struct node *prev;
    struct node *next;  };
```

and ptr is a pointer pointing to one of the internal nodes of a doubly linked list, then write the required statements to delete the node to which ptr is pointing to without using any other pointer variable.

Ans:

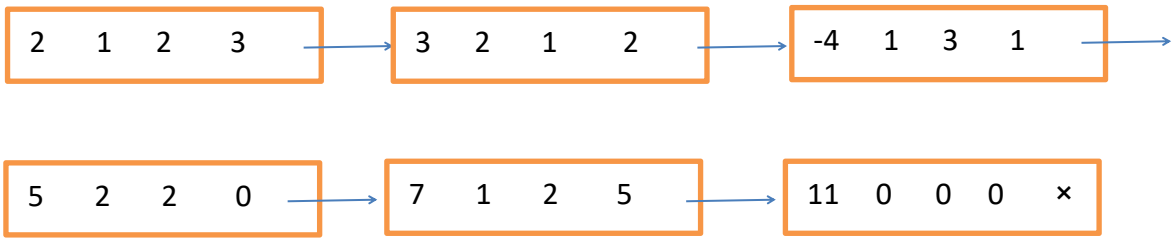
```
ptr->prev->next=ptr->next;
ptr->next-prev=ptr->prev;
free(ptr);
```

OR

```
ptr->next-prev=ptr->prev;
ptr->prev->next=ptr->next;
```

free(ptr);

d) How a singly linked list can be used to represent the following polynomial:
$P(x)=2xy^2z^3+ 3x^2yz^2- 4xy^3z+5x^2y^2+7xy^2z^5+11$
Ans:




e) A stack is implemented using an array declared in C: st[N] and an integer variable: pos. The pseudo code for the push() and pop() operations of the stack are as follow:

```
push(x){
    st[pos]=x;
    pos=pos-1;
}
pop() {
    pos=pos+1;
    return(st[pos]);
}
```

Find the initialization of integer variable: pos for an empty stack with a maximum capacity of N elements for the above implementation.
Ans:
pos = N-1

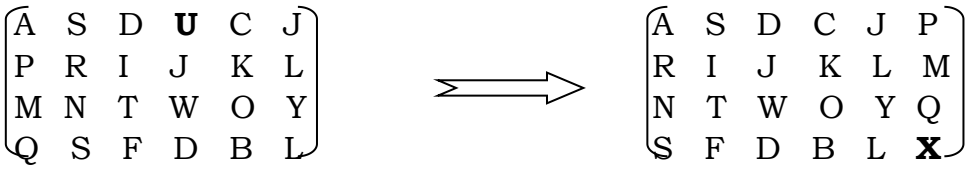
2.

a) Write a pseudo code/ C-function to sort the elements present at the even index of an array in ascending order without using any additional data structure. 2
Ans:

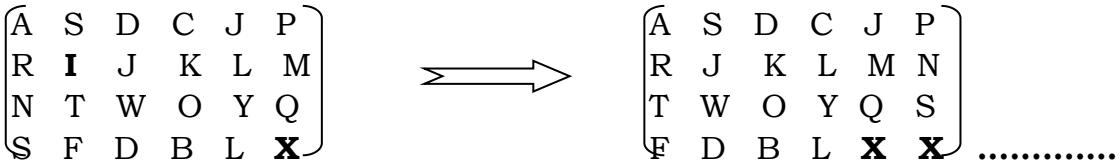
```
void sortEvenPos(int arr[], int n) {
        int i, j, tmp;
        for(i=0; i<n; i+=2)
                for(j=i+2; j<n; j+=2)
                        if(arr[i] > arr[j]) {
                                tmp = arr[i];
                                arr[i] = arr[j];
                                arr[j] = tmp;
                        }
}
```

b) Create two dimensional array of M×N dynamically containing English alphabets (capital letters) except 'X'. Write a pseudo code to delete ALL the vowels present in the matrix. Once any vowel found in a row and deleted, the elements will be shifted from the same row as well as the next rows to fill the deleted element. The empty space at last will be replaced by alphabet 'X'. 3

E. g. For deletion of the first vowel (U),




For deletion of the second vowel (I),




.....

Ans: (Step marking can be done depending on correctness)

```
#include<stdio.h>
#define m 4
#define n 6
char a[4][6]={{'A', 'S', 'D', 'U', 'C', 'J'},
 {'P', 'R', 'I', 'J', 'K', 'L'},
 {'M', 'N', 'T', 'W', 'O', 'Y'},
 {'Q', 'S', 'F', 'D', 'B', 'L'}};
int main() {
        char *p=&a[0][0];
        int total=m*n;
        int i, j;
        for(i=0; i<total; i++) {
                if(p[i]=='A'|| p[i]=='E'|| p[i]=='I' || p[i]=='O' || p[i]=='U') {
                        for(j=i+1;j<total;j++)
                                p[j-1]=p[j];
                        total--;
                        p[total]='X';
                }
        }
        for(i=0; i<4; i++) {
                for(j=0; j<6; j++)
                        printf("%c   ", a[i][j]);
                printf("\n");
        }
        return 0;
}
```

3.

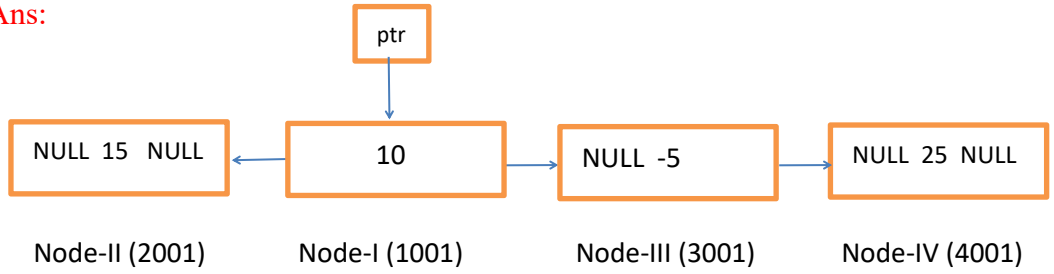
a)

| Node | Data | Prev Address | Next Address | Node Address |
|---|---|---|---|---|
| I | 10 | 2001 | 3001 | 1001 |
| II | 15 | NULL | NULL | 2001 |
| III | -5 | NULL | 4001 | 3001 |
| IV | 25 | NULL | NULL | 4001 |

2

The above table is a representation of doubly linked list where the nodes are declared in C as: struct node { int data; struct node *prev; struct node *next; }; and connected as per the data shown in the table. Draw the linked list. Also, write C-like statements to replace the values of Node-III and Node-IV by adding the value of Node-I to these node values and replace the value of Node-II by subtracting the value Node-I from it. Consider the base pointer ptr, which holds the address of Node-I. Do not use any other pointer variable.

Ans:




```
ptr->next->data += ptr->data;
ptr->next->next->data += ptr->data;
ptr->prev->data -= ptr->data;
```

b) The following data are stored in a singly linked list in the order shown below: 3
1->2->3->4->5->6->7->8->9->10
Write a pseudo code/ C-function to delete all odd numbers from the list.

Ans:

```
void deleteOddValueNodes(struct Node **head) {
        struct Node *temp = *head, *prev;
        while(temp != NULL && temp->data % 2 == 1) {
                *head = temp->next;
                free(temp);
                temp = *head;
        }
        while(temp != NULL) {
                while(temp != NULL && temp->data % 2 != 0) {
                        prev = temp;
                        temp = temp->next;
                }
                if(temp == NULL)
                        return;
                prev->next = temp->next;
                free(temp);
                temp = prev->next;
        }
}
```

4. Explain sparse matrix and its representations. Write a pseudo code/ C-function to display triplet of transpose a sparse matrix with a suitable example. 5

Ans: (Explanation 2 marks, Algorithm 3 marks)

If most of the elements in a matrix are zero then the matrix is called a sparse matrix. It is wasteful to store the zeroes in the matrix since they do not affect the results of the

computation. Sparse Matrix Representations can be done in following two common representations:
Array representation
Linked list representation

Method 1: Using Arrays:
2D array is used to represent a sparse matrix in which there are three columns named as:

- Row: Index of row, where non-zero element is located
- Column: Index of column, where non-zero element is located
- Value: Value of the non zero element located at index – (row,column)

Method 2: Using Linked Lists
In linked list, each node has four fields. These four fields are defined as:

- Row: Index of row, where non-zero element is located
- Column: Index of column, where non-zero element is located
- Value: Value of the non zero element located at index – (row,column)
- Next node: Address of the next node

Transpose:

```
void transpose(int b1[][3], int b2[][3]) {
        int i, j, k, n;
        b2[0][0]=b1[0][1];
        b2[0][1]=b1[0][0];
        b2[0][2]=b1[0][2];
        k=1;
        n=b1[0][2];
        for(i=0;i<b1[0][1];i++)
                for(j=1;j<=n;j++)
                        if(i==b1[j][1]) {
                                b2[k][0]=i;
                                b2[k][1]=b1[j][0];
                                b2[k][2]=b1[j][2];
                                k++;
                        }
}
```

5. Write an algorithm/ pseudo code/ C-function to convert an expression in infix notation to its equivalent prefix expression. Convert the infix expression: (A+B)+C-(D-E)^F to prefix equivalent mentioning each step of the algorithm. 5

Ans: (Algorithm 3 marks, problem 2 marks)

```
Function InfixtoPrefix(infixExpr)
begin
        infixExpr = reverse(infixExpr)
        empty prefixExpr
        for(i=0; i<length(infixExpr); i++)
                if(infixExpr[i] == operand)
                        append infixExpr[i] to prefixExpr
                else if(infixExpr[i] == ')')
```

```
                push(infixExpr[i])
            else if(infixExpr[i] == '(')
                while(top() != ')')
                    append(top()) to prefixExpr
                    pop()
            else if(infixExpr[i] == operator)
                while(!empty() && prec(top()) > prec(ch))
                    append top() to prefixList
                    pop()
                push(ch);
        while(!empty())
            append top() to prefixExpr
            pop()
        prefixExpr = reverse(prefixExpr)
        return
end
```

| | |
|---|---|
| Infix Expr | (A+B)+C-(D-E)^F |
| Reversed Expr | F^)E-D(-C+)B+A( |

| Scanned Token | Stack Status | Expression |
|---|---|---|
| F | | F |
| ^ | ^ | F |
| ) | ^ ) | F |
| E | ^ ) | F E |
| - | ^ ) - | F E |
| D | ^ ) - | F E D |
| ( | ^ | F E D - |
| - | - | F E D - ^ |
| C | - | F E D - ^ C |
| + | - + | F E D - ^ C |
| ) | - + ) | F E D - ^ C |
| B | - + ) | F E D - ^ C B |
| + | - + ) + | F E D - ^ C B |
| A | - + ) + | F E D - ^ C B A |
| ( | - + | F E D - ^ C B A + |
| | | F E D - ^ C B A + + - |
| Prefix Expr | - + + A B C ^ - D E F | |