Mid Semester Examination
School of Computer Engineering
KIIT UNIVERSITY, BHUBANESWAR

Time: 2hrs **Full Mark: 25**

[ANSWER ANY FIVE QUESTIONS INCLUDING Q.NO.-1]

1. Answer all the questions [1 **X 5**]

a. Define the term Data Structure and ADT with example. List any four applications of data structure.

Ans: **Data structure definition (0.5)**
ADT Definition (0.5)

A **data structure** is a specialized format for organizing and storing **data**.

An **abstract data type (ADT)** is a mathematical model for data types where a data type is defined by its behavior from the point of view of a user of the data.

Data structure includes arrays, link lists, trees, etc.

b. Find the time complexity of the following code snippet:

```
void function(int n){
        int i,j,k;
        for(i=n/2; i<=n; i++)
                for(j=1; j + n/2<=n; j= j++)
                        for(k=1; k<=n; k= k * 2)
                                count++; }
```

Ans: **$O(n^2 * \log n)$ or $O(n^2/4 * \log n)$. In both these type of answers, we should give the marks because students do not know mathematical computation of time complexity.**

c. Compare and contrast Array ADT and Linked List ADT.

Ans: **pros and cons of both the term must be specified. WE can expect different kinds of answer.**
Only definition (0.5)
comparison(0.5)

d. Implement a stack ADT.

Ans:

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

```
typrdef struct {
        struct node *top;
}STACK;
```

One can also do the STACK implementation using an array.

```
#define MAX 100

typrdef struct {

        int data[MAX];
        int top;
}STACK;
```

e. Find the address of the element at index 65 and 93 of a two dimensional array, whose row index range from 35 to 93 and column index range from 75 to 99. The two dimensional address is stored in column major order and the base address of the matrix is 1000.

Ans: 1000+{(65-35)+(93-75)*(93-35+1)}*size

2.

Write an algorithm to convert an infix expression into its equivalent postfix expression. Explain the execution of the algorithm using the following expression. **[5]**

((A + B) * C - (D - E) ^ (F + G))

Ans: Algorithm (3 marks)

Conversion of above infix expression to postfix expression step-by-step (2 marks)

Direct answer of above expression conversion (0.5 marks)

((A + B) * C – (D- E) ^ (F + G))

Symbol	Stack	Out
(	(	
(	((	
A	((	A
+	((+	A
B	((+	AB
)	(	AB+
*	(*	AB+
C	(*	AB+C
-	(-	AB+C*
(	(-(	AB+C*
D	(-(	AB+C*D

-	(-(-	AB+C*D
E	(-(-	AB+C*DE
)	(-	AB+C*DE-
^	(-^	AB+C*DE-
(	(-^(	AB+C*DE-
F	(-^(	AB+C*DE-F
+	(-^(+	AB+C*DE-F
G	(-^(+	AB+C*DE-F
)	(-^	AB+C*DE-F+
)		AB+C*DE-F+^-

3.

a) How do we represent a polynomial expression using linked list? Write a pseudo code to add two polynomial having two numbers of unknown variables. **[3]**
(*e.g.* : $4x^2y^3-3xy+x-5y+7$)

Ans: <u>polynomial expression representation using linked list (1 marks)</u>
<u>pseudo code to add two polynomial (2 arks)</u>

<u>polynomial expression representation using linked list</u>

(4,2,3,)->(-3,1,1,)->(1,1,0,)->(5,0,1,)->(7,0,0,)

<u>Sample Code</u>

```
struct node{
        int cof;
        int xexp;
        int yexp;
        struct node *next;
};

void add(struct node *head1, struct node *head2)
{
        struct node *ptr1=head1,*ptr2=head2,*prev;
        while(ptr1->next!=NULL)
                ptr1=ptr1->next;
        ptr1->next=head2;

        for(ptr1=head1;ptr1->next!=NULL;ptr1=ptr1->next)
```

```
{
        prev=ptr1;
        for(ptr2=ptr1->next;ptr2!=NULL;ptr2=ptr2->next)
        {
                if((ptr1->xexp==ptr2->xexp) && (ptr1->yexp==ptr2->yexp))
                {
                                ptr1->cof=ptr1->cof+ptr2->cof;
                                prev->next=ptr2->next;
                                free(ptr2);
                                ptr2=prev;
                }
                prev=ptr2;
        }
        if(ptr->next==NULL)
                break;
}
display(head1);
                                        }
```

OR

Write a pseudo code to merge two sorted linked list. **[3]**

Ans: pseudo code to merge two sorted linked list (3 marks)

Sample Code

```
mergelist(struct node **h1, struct node **h2){
        struct node *ptr1,*head3=NULL;
      while(*h1!=NULL && *h2!=NULL){
          if((*h1)->data<(*h2)->data){
                  ptr1=*h1;
                (*h1)=(*h1)->next;
                insert(ptr1,&head3)
            }else if((*h1)->data>(*h2)->data){
                  ptr1=*h2;
                (*h2)=(*h2)->next;
               insert(ptr1,&head3);
           }
      }
        while(*h1!=NULL){
                  ptr1=*h1;
                (*h1)=(*h1)->next;
               insert(ptr1,&head3);
      }
    while(*h2 != NULL){
```

```
            ptr1=*h2;
            (*h2)=(*h2)->next;
            insert(ptr1,&head3);
        }


    }
```

b) Write a pseudo code to check whether a given postfix expression is valid or not. **[2]**

Ans: Algorithm (2 marks)

Using STACK one can verify

i) Iterate through each token present in the postfix expression.

ii) If an operand comes push into the STACK.

Iii) If an operator comes, POP elements from the STACK. If sufficient operand is not available in the STACK then the postfix expression is invalid. If sufficient operands are available then perform the operation and push the result again into the STACK.

iv) If at the end the STACK contains more than one element then the postfix expression is invalid.

4.

a) Write a program to copy the elements of one stack to another stack without changing the order. **[3+2]**

[**Note**: The program should not take the help of any additional data structures]

Ans: Algorithm (3 marks)

```
void copySTACK(STACK *s1,STACK *s2){
    int b;
    if(isEMPTY(s1))
        return;
    pop(s1,&b);
    copySTACK(s1,s2);
    push(s2,b);
}
```

b) What is a sparse matrix? How do we represent a sparse matrix?

Ans: sparse matrix definition (1 marks)

Representation of sparse matrix (1 arks)

5.

a) Write a function to discard the common elements present in between two linked list from the original linked list. **[3]**

Ans: Algorithm (3 marks)

```
void discardfun(struct node **h1, struct node **h2){
struct node *ptr1=*h1,*ptr,*ptr2;
```

```
while(ptr1!=NULL){
for(ptr2=*h2;ptr2!=NULL;ptr2=ptr2->next){
if(ptr1->data==ptr2->data){
ptr=ptr1;
ptr1=ptr1->next;
delete(ptr,h1);
break;
}
}
}
}
```

b) Write a pseudo code to reverse a single linked list by changing the required links. **[2]**

Ans: Algorithm (2 marks)

```
void reverselink(struct node **h){

struct node *ptr1=*h,*ptr2;
if(*h==NULL)
        return;
ptr2=ptr1->next;
while(ptr2!=NULL){
ptr1->next=ptr2->next;
ptr2->next=*h;
*h=ptr2;
ptr2=ptr1->next;
}
}
```

OR

Define Queue ADT. Write a program to implement insertion, deletion and traversing operation on circular queue ADT. **[5]**

Ans: Queue ADT (1 marks)
insertion (1 marks)
deletion (1 marks)
traversal (2 marks)

6.

Write a pseudo code to traverse a NXN 2D-array in row major order. During traversing, if the current traversed element is greater than the previously traversed element, then delete the current element and shift the remaining elements of the matrix. The last blank place should be filled with zero. The sequence of execution is shown below:

3	6	4
2	7	1
8	5	9

3	**6**	4
2	7	1
8	5	9

3	**4**	2
7	1	8
5	9	**0**

3	**2**	7
1	8	5
9	**0**	**0**

(I) **(II)** **(III)** **(IV)**

[The elements will be traversed in row major order i.e.: 3, 6, 4, 2, 7, 1, 8, 5 and 9]

(I): Input Matrix, First element '3' is traversed.

(II): Current element traversed is '6' which is greater than previously scanned element '3'. Hence '6' is deleted and elements are shifted and finally last position is filled with zero.

(III): Current element traversed is '4' which is greater than previously scanned element '3'. Hence '4' is deleted and elements are shifted and finally last position is filled with zero.

(IV): Current element traversed is '2' which is less than previously scanned is '3'. Hence there will be no shifting of elements.

In the similar fashion, rest of the steps will be executed. **[5]**

Ans: <u>Algorithm (5 marks)</u>

```
#include<stdio.h>
#define n 3
int main()
{
        int a[n][n],i,j,*ptr=&a[0][0];;
        printf("enter teh elements of matrix:");
        for(i=0;i<n;i++)
                for(j=0;j<n;j++)
                        scanf("%d",&a[i][j]);
        int total=n * n;
        for(i=1;i<total;i++){
                if(ptr[i]>ptr[i-1])
                {
                        for(j=i+1;j<total;j++)
                                ptr[j-1]=ptr[j];
                        i--;
                        total--;
                        ptr[total]=0;
                }
        }
        return 0;
}
```
