CMR INSTITUTE OF TECHNOLOGY

Solution/Model Answer of IAT-2 (Oct 2018)
Data Structures & Applications (17CS33)
III Sem – CSE
Dr. P. N. Singh, Professor(CSE)

---

1. **Write a C program to merge 2 sorted array of different dimensions in a 3rd array.** **10**

Ans:

/*Merging 2 sorted arrays of different sizes in a 3rd array */

```
#include <stdio.h>

main()
{
  int a[7]={22,33,44,55,666,777,888,};
  int b[5]={100,200,300,890,900}, c[12], ax,bx,cx;
  printf("Array a:\n ");
  for(ax=0;ax<7;ax++)
    printf("%5d",a[ax]);

  printf("\nArray b:\n ");
  for(bx=0;bx<5;bx++)
    printf("%5d",b[bx]);
  ax=bx=cx=0;
  while(ax<7 && bx<5)
  {
     if (a[ax] < b[bx]) { c[cx]=a[ax]; ax++; }
     else { c[cx] = b[bx]; bx++; }
     cx++;
  }
  /* for rest of the elements */
  while (ax < 7) { c[cx]=a[ax]; ax++; cx++; }
  while (bx < 5) { c[cx]=b[bx]; bx++; cx++; }

  printf ("\nNow merged and sorted array c:\n");
  for(cx=0;cx<12;cx++)
   printf("%5d",c[cx]);
return (0);
 }
```

```
Output:
Array a:
```

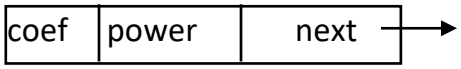
```
    22   33   44   55  666  777  888
Array b:
   100  200  300  890  900
Now merged and sorted array c:
   22   33   44   55  100  200  300  666  777  888  890  900
```

2. a. What is a polynomial list? Define structure & represent the Multivariate polynomial $3x^3yz + 6x^2y^2z+2xy^5z-2xyz^3-4yz^5$ in a linked list diagram. 5

Ans: A polynomial list has two members in structure i.e. coefficient and exponentiation (power)

/* node structure for linked representation of a polynomial */

```
typedef struct poly
{
        int coef;
        int power;
        struct poly *next;
 }poly *p1, *p2, *p3;
```



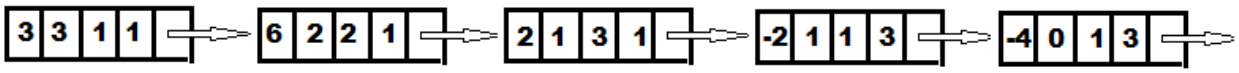
List Representation:

Multivariate polynomial: A polynomial in more than one indeterminate is known as multivariate polynomial.

Proposed Structure for Multivariate Polynomial

struct mvpoly { int coef,xexpo,yexpo,zexpo };

Representing Multivariate polynomial : $3x^3yz + 6x^2y^2z+2xy^5z-2xyz^3-4yz^5$



b. Write C function to add two polynomials 5

Ans:

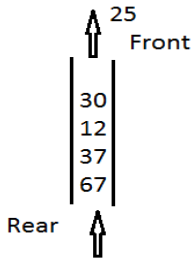
```
void addpoly(node *p1, node *p2, node **p3)
 {
  int po;
  float co;
  while ( ( p1 != (node *) NULL &&    p2 != (node *) NUL
  {
```

```
    if(p1->power > p2->power)  { co = p1->coeff; po = p1->power; p1 = p1->next;  }
   else
    if ( p1->power< p2->power) { co = p2->coeff;  po = p2->power; p2 = p2 -> next;  }
   else { co = p1 -> coeff + p2 -> coeff; po = p1 -> power; p1 = p1 -> next; p2 = p2 -> next; }
  if (co != 0)     addnode(p3, co, po);
}

if (p1 == (node *) NULL)
{
 for (; p2 != (node *) NULL; p2 = p2 -> next)
   addnode(p3, p2->coeff,p2->power);
}
            else if (p2 == (node *) NULL)
            {
 for (; p1 != (node *) NULL; p1 = p1 -> next)
               addnode(p3, p1->coeff,p1->power);
            }

  }
```

3. What is benefit of Circular Queue over linear queue? Write insertq() and deleteq() functions for circular queue. **10**

Ans:

Queue is FIFO (first in first out data structure). Items are processed and deleted according to their sequence of arrival in the queue. "Items are inserted from rear and deleted from front end.

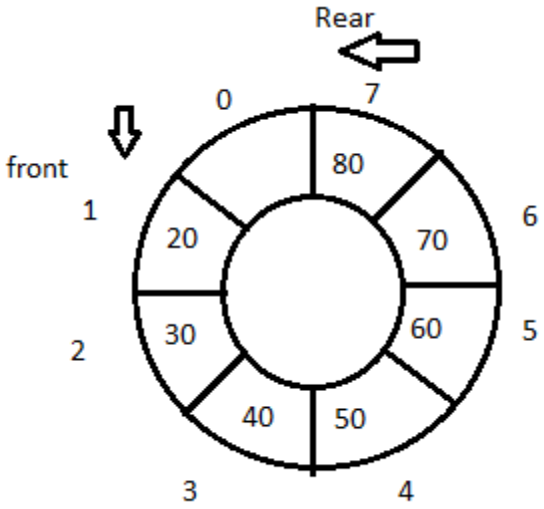


A linear queue of size 5

Disadvantages of linear queue:

- Items cannot be inserted from front end even the space is available
- Items cannot be deleted from rear or middle

This can be solved using a circular queue where front and rear are adjacent when queue is full. If any item is deleted then variable rear can be used to insert item to utilize it.



A circular Queue of size 8 (0 to 7)

One item is deleted from front end and queue[0] is vacant. Now front = 1 and rear =7 so by rear variable it can be occupied

```
void insertq ( int arr[], int item )
{
   if ( ( rear == MAX - 1 && front == 0 ) || (  rear + 1 == front ) )
   {
      puts("\nCircular Queue is full" );return ;
   }
   rear=(rear+1)%MAX;    arr[rear] = item ;
   if ( front == -1 ) front = 0 ; /* when queue has been inserted an item */
}

int deleteq(int arr[] )
{
   int data ;
   if ( front == -1 ) {  puts("\nQueue is empty") ;      return (NULL) ;  }
   data = arr[front] ; arr[front] = 0 ;
   if ( front == rear ) {        front = -1 ; rear = -1; } /* when all elements are deleted */
   else       front=(front+1)%MAX;
   return (data) ;
}
```

4. Develop a C program to reverse a Singly Linked List. **10**

Ans:

```
/* reversing linked list */
#include <stdio.h>
#include <stdlib.h>
```

```
struct list { int number;              struct list *next; };
typedef struct list node;
void create(node *list)
 {
  printf("Input a number -> ( 0 to end): ");
  scanf("%d",&list->number);
   if (list->number == 0)
     list->next = NULL;
   else
    { list->next = (node *)malloc(sizeof(node));
      create(list->next); }
  return;
  }

 void print(node *list1)
 {
  if (list1->next != NULL)
   { printf("| %d | --->",list1->number);
     print(list1->next);
    }
    return;
  }

  /* function to reverse a linked list */
  void rev(node *list)
  {
    node *prevnode, *tempnode, *nextnode;
    nextnode=list->next;
    prevnode=list;
    prevnode->next=NULL;

    while(nextnode->next!=NULL)
      {
       tempnode=nextnode;
       nextnode=nextnode->next; /* again first storing the address of next node*/
       tempnode->next=prevnode; /* and now altering the path  - Dr. P. N. Singh*/
       prevnode=tempnode;

       }
       /* printing reverse list */
       while(prevnode!=NULL)
       { printf("| %d | --->",prevnode->number);
         prevnode=prevnode->next;
       }
    }
```

```
int main()
{
  node *head;
  head = (node *)malloc(sizeof(node));
  create(head);
  puts("The linked list\n");
  print(head);
  puts("\nThe reversed linked list\n");
  rev(head); /* calling function to reverse */
  return (0);
}
```

5. a. Demonstrate with logic to insert a new node between 2 nodes of Doubly Linked List. How many pointers are to be modified? Justify. 5

Ans:

```
/* insert a node between two nodes (after specified node) */
void d_addafter ( struct dnode *q, int loc, int num )
{
    struct dnode *temp ;
    int i ;

    /* skip to desired portion */
    for ( i = 0 ; i < loc ; i++ )
    {
        q = q -> next ;
        /* if end of linked list is encountered */
        if ( q == NULL )
        {
            printf ( "\nThere are less than %d elements", loc );
            return ;
        }
    }

    /* insert new node */
    q = q -> prev ;
    temp = malloc ( sizeof ( struct dnode ) ) ;
    temp -> data = num ;
    temp -> prev = q ;
    temp -> next = q -> next ;
    temp -> next -> prev = temp ;
    q -> next = temp ;
}
```

Pointers to be modified (inserting node between two nodes): given in above example
If newnode is inserted after q and before q->next
newnode -> prev = q ;

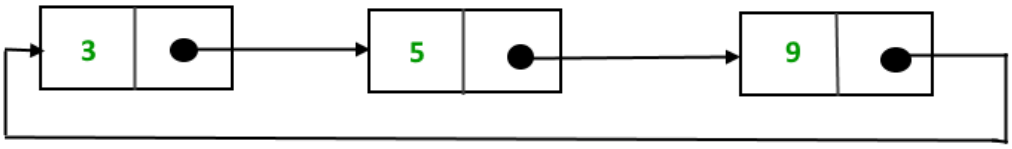
```
newnode -> next = q -> next ;
newnode -> next -> prev = newnode ;
q -> next = newnode ;
```

prev and next of newnode, next of previous node & prev of next node, so total 4 pointers will be modified.

5. b. Write a C function to insert a node at the end of a Single Circular Linked List. **5**

Ans:

Singly circular linked list where pointer of last node keeps address of the first node.



Function to add a node at the end of Singly circular linked list

```
struct Node *addEnd(struct Node *last, int data)
{
  if (last == NULL)
     return addToEmpty(last, data);

  // Creating a node dynamically.
  struct Node *temp =
        (struct Node *)malloc(sizeof(struct Node));

  // Assigning the data.
  temp -> data = data;

  // Adjusting the links.
  temp -> next = last -> next;
  last -> next = temp;
  last = temp;

  return last;
}
```

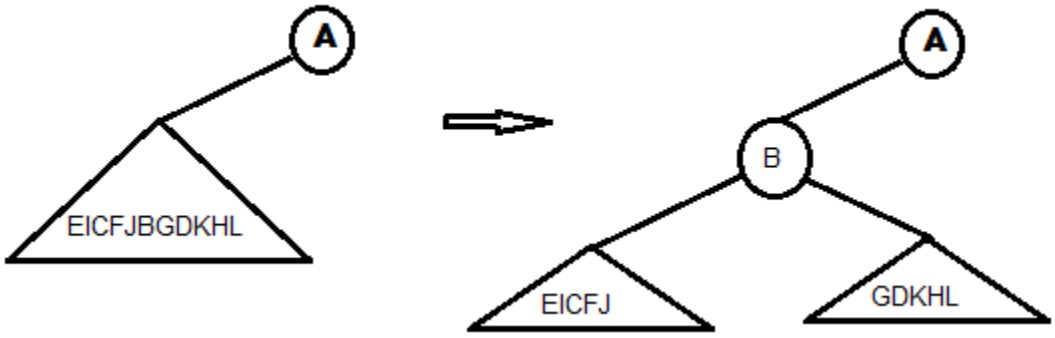
6. a. Draw a Binary Tree by following given traversal. **5**

Pre-Order : A B C E I F J D G H K L
In-Order : E I C F J B G D K H L A

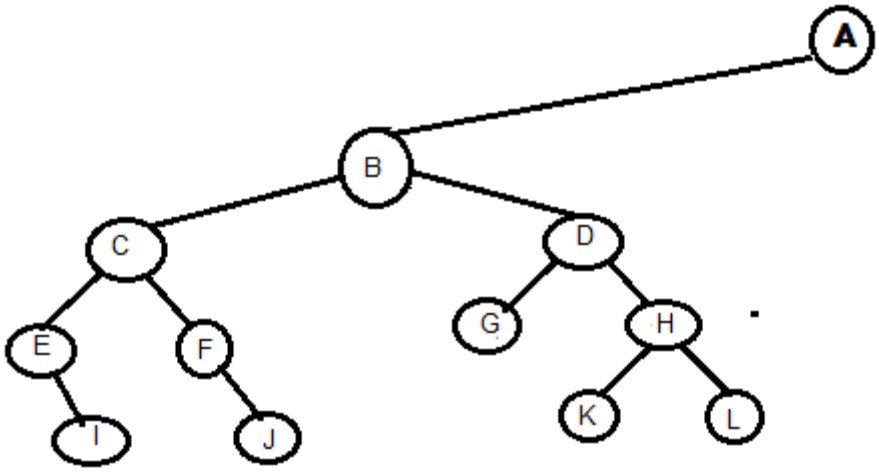
Ans:

In pre-order traversal sequence is “Root-Left-Right” so A is the root node
In in-order traversal sequence is “Left-Root-Right” and here A is shown as last node, so all rest of the nodes are left to A.

In pre-order next node is visited B, so B is left to A.
In in-order E,I,C,F and J are left to B and G,D,K,H & L must be right to B.



Accordingly seeing the sequence in pre-order and finding nodes left and right in in-order the Binary tree will be like this which satisfies both traversals.

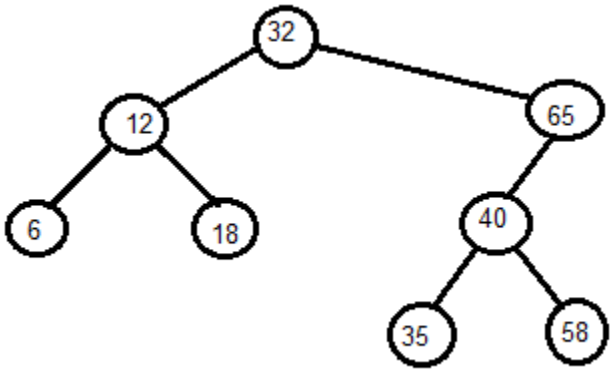


6. b. Insert 32,12,6,18,65,40,58,35 in a Binary Search Tree & write In-order, Pre-Order and Post Order traversals. 5

Ans:

In a Binary Search Tree nodes are inserted as leaf node following Binary Search(value less than parent goes to left and value greater than parent goes to right)

After inserting all the nodes:



In-order traversal: (“Left-root-right”) : 6,12,18,32,35,40,58,65

Pre-order Traversal: (“Root-Left-Right”): 32,12,6,18,65,40,35,58

Post order traversal: (“left-Right-Root”): 6,18,12,35,58,40,65,32

7. Develop recursive functions of In-order, Pre-order & Post Order traversals for a Binary Search Tree. 10

Ans:

In-Order Traversal of a Binary Tree:

- Visit Left Sub-tree
- Visit the Root Node
- Visit Right Subtree

Routine/Recursive Function for In-order Traversal (Assuming that Binary tree structure has 3 members data, left and right pointers to the structure):

```
void inorder(TREE *bintree)
  {
   if(tree)
    {
                    preorder(bintree->left);
                    printf(" %d",bintree->data);
                    preorder(bintree->right);
    }
  }
```

Pre-Order Traversal of a Binary Tree:

- Visit the Root Node
- Visit Left Sub-tree
- Visit Right Subtree

Routine/Recursive Function for Pre-order Traversal (Assuming that Binary tree structure has 3 members data, left and right pointers to the structure):

```
void preorder(TREE *bintree)
  {
   if(tree)
    {
                    printf(" %d",bintree->data);
                    preorder(bintree->left);
                    preorder(bintree->right);
    }
  }
```

Post-Order Traversal of a Binary Tree:

- Visit Left Sub-tree
- Visit Right Subtree
- Visit the root node

Routine/Recursive Function for Post-order Traversal (Assuming that Binary tree structure has 3 members data, left and right pointers to the structure):

```
void postorder(TREE *bintree)
  {
   if(tree)
    {
                    postorder(bintree->left);
                    postorder(bintree->right);
                    printf(" %d",bintree->data);
    }
  }
```

8a. Write a C program to implement Priority Queue. **5**

Ans:

```
/* Priority queue program – Dr. P. N. Singh, Professor(CSE) */
#include <stdio.h>
#include <string.h>
#define MAX 5
struct data {        char job[10] ;    int prno ; };
int front, rear ;
struct data d[MAX];

void pque( )
{
   int i;     front = rear = -1 ;
```

```
    for ( i = 0 ; i < MAX ; i++ )  { strcpy ( d[i].job, '\0' ) ; d[i].prno = 0 ;  }
}

void add ( struct data dt )
{
    struct data temp;
    int i,j;
    if ( rear == MAX - 1 )      { printf("\nQueue is full") ;         return ; }
    rear++ ;            d[rear] = dt ;
    if ( front == -1 )  front = 0 ;
    for ( i = front ; i <= rear ; i++ )
            for ( j = i + 1 ; j <= rear ; j++ )
                    if ( d[i].prno > d[j].prno ) { temp = d[i] ;   d[i] = d[j] ; d[j] = temp ;  }
}

struct data del()
{
    struct data  t ;   strcpy ( t.job, "" ) ;         t.prno = 0 ;
    if ( front == -1 )  { printf("\nQueue is Empty\n");  return t ; }
    t = d[front] ;      d[front] = t ;
    if ( front == rear )  front = rear = -1 ;       else                front++ ;
    return  t ;
}

main( )
{

    struct data dt,temp ;
    int i ;
    pque();
    printf("Enter Job description and its priority number:");
    printf("Lower the priority number, higher the priority:") ;
    printf("\nJob Description    Priority\n");

    for ( i = 0 ; i < MAX ; i++ ) { scanf("%s%d",dt.job,&dt.prno) ; add ( dt ) ;     }
    printf("\n");
    printf("Process jobs priority wise\n");

    printf("Job    Priority\n");
    for ( i = 0 ; i < MAX ; i++ ) {          temp  =  del( ) ;  printf("%s    %d\n",temp.job,temp.prno); }
    printf("\n");
  return 0;
}
```

8b. What is a "Dequeue"? Explain input restricted and output restricted dequeue. **5**

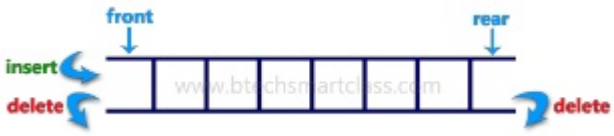
Ans:

A double-ended queue, or **deque**, supports insertion and deletion from the front and rear

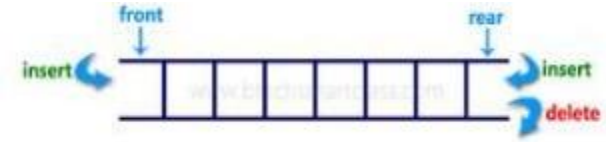
The possible operation performed on deque:

- Add an element at the rear end
- Add an element at the front end
- Delete an element from the front end
- Delete an element from the rear end

- An input-restricted deque is one where deletion can be made from both ends, but insertion can be made at one end only.



- An output-restricted deque is one where insertion can be made at both ends, but deletion can be made from one end only



-