Q.2 a. Write a function that computes x^y using Recursion.

Answer:
The property that x^y is simply a product of x and x^(y-1). For example , 5^4= 5 * 5^3. The recursive definition of x^y can be represented as shown below:-

power(x,y) = 1 if y=0
x*power(x,y-1) if y > 0

The required function is given below:-

```
/* function that computes x^y */
        long power ( int x , int y)
{
        If( y==0)
                return(1);
        else
                return( x * power( x , y-1 ));
}
```

b. Describe local scope of variables using a suitable example.

Answer:
The range of code in a program over which a variable has a meaning is called as scope of the variable. In simple words the scope of a variable decides as to how it can be accessed by the different parts of the program.

In fact there are three types of scopes in C

1. Local scope 2. Function scope 3. File scope

Local Scope: A block in C is surrounded by a pair of curly braces i.e. '{' and '}' . So a variable can be declared inside the block.Such variables, called as local, can be referenced only within the body of the block. When the control of program execution reaches the opening brace, these variables come in to existence and get destroyed as soon as the control leaves the block through the closing brace . It may be noted that the variables are available to all the blocks enclosed by the block in which they have been declared.

Example:

```
while (flag)
{       int x;
        if (i > x)
        {       int j;    }}
```

The scope of variable 'x' is the while block and that of variable x can also be referenced to in the if block because this block is completely enclosed in the while block. On the other hand , the scope of variable 'j' is only the 'if' block. Since a function is a block in itself, all the variables declared in, it have the local scope.The formal parameters declared in a function have the local scope.

c. When is register allocation done?

Answer: Refer Page 119 of Text Book

Q.3 a. How are the Structures defined and initialized? Briefly explain it with an appropriate example.

Answer:

A structure can also be initialized in the way as any as any other data type. A structure to be initialized must be either static or external. The initialization of structures is done at object declaration step. For example: the day, month and year of the structure data having an object 'a' can be initialized as:

static struct date a = { 31, 12, 1998];

The compiler will automatically assign these initialized values to each of the fields, such as:

day =31 month=12 year=1998

When a structure variable is declared , its data members are not initialized and therefore contain undefined values.Similarly to arrays, the structure variables can also be initialized .The values are assigned to the members of the structure in the order of their appearance i.e. first value is assigned to the first member, second to second and so on.

b. Write a program to initialize a structure of a student having rollno, gender, height, weight and display the contents using structure pointer.

Answer:

```
#include< stdio.h>
#include<conio.h>
void main()
{
struct student
{ long int rollno;
Char sex;
float height, weight;
} struct student data;
struct student *ptr1;
clrscr();
ptr1= &data;
ptr1->rollno=10;
ptr1->sex='m';
ptr1->height=5.10;
ptr1->weight=65;
printf(" The entered contents are :\n");
printf("%d\n", ptr1->rollno);
printf("%c\n", ptr1->sex);
printf("%f\n", ptr1->height);
printf("%f\n", ptr1->weight);
getch();
}
```

The output of the program is :-

```
The entered contents
are:
10
M
5.10
65
```

Q.4 a. Define an Array. Write a program which reads a list of ten numbers and prints the list in reverse order.

Answer:
Definition : An array is a collection of identical data objects or homogeneous type of data objects which are stored in consecutive memory locations under a common heading or a variable name.

The program for this problem is :-

```
main()
{
int list[20],n,i;
printf("enter the number of integer elements in the list");
scanf("%d",&n);
printf("\n enter the elements");
for (i=0;i<n;i++)
{
printf("\n enter number");
scanf("%d",&list[i]);   }
printf("\n the list in reverse order");
for (i=n-1;I>=0;i--)
printf("%d",list[i]);
}
```

b. (i) Can an array be assigned to another of same size and type? For example if two arrays A[5] and B[5] are available of the same type, what would be the effect of following statement? A=B

(ii) Write a program to merge two sorted arrays in third one.

Answer:
(i) No, an array can not be assigned to another. The arrays are designed to work in component by component fashion. The above given assignment statement would result in syntax area.

(ii)

```
void main()
    {   int p,q,m,n,c;
    int array1[100],int array2[100],array3[100];
    puts("enter number of elements in first array");
    scanf("%d",&p);
    puts("enter elements of the first sorted array");
    for(m=0;m<=p;m++)
    { scanf("%d",&array1[m]);}
    puts("enter number of elements in second array");
    scanf("%d",&q);
    puts("enter elements of the first sorted array");
    for(n=0;n<=q;n++)
    { scanf("%d",&array2[n]);}
    c=0;m=0;n=0;
    while((m<q)&&(n<q))
    {
    If(array1[m]<=array2[n])
    array3[c]=array1[m+1];
    else
    array3[c]=array2[n++];
    c++;
    }
    while(m<p)
    {
    array3[c]=array1[m];
    c++;
```

```
m++;
}
while(n<q)
{
array3[c]=array2[n];
c++;
n++;
}
puts("merged array in ascending order");
for(m=0;m<=c;m++)
printf("%d",array3[m]);
}
```

Q.5 a. Discuss in detail the various methods of Stack Implementation.

Answer: Stack can be implemented in two ways:-
(i) Static implementation (ii) Dynamic implementation

(i) Static implementation:- uses arrays to create stack.Static implementation though a very simple technique but is not a flexible way of creation, as the size of stack has to be declared during program design, after that the size cannot be varied.Moreover, static implementation is not too efficient with respect to memory utilization.As the declaration of array for stack is done, before the start of the operation, now if there are too few elements to be stored in the stack, the statically allocated memory will be wasted , on the other hand, if there are more number of elements to be stored in the stack then size of the array cannot be increased to increase the capacity, so that it can accomodate new elements.

(ii) Dynamic implementation:- is also called linked list representation and uses pointers to implement the stack type of data structure.Single linked list structure is sufficient to represent any stack.Here, the info, field is for the item, and next field is as usual to point to the next item.

'top' is a pointer variable which will point to the top of the stack. To initialize a stack, one simply set its top to NULL. So declaration is :-

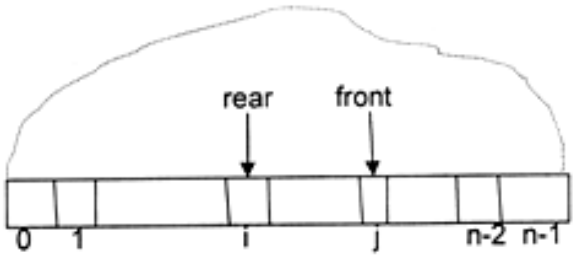
```
struct node
{
    int info;
    struct node *next;
};
typedef struct node stack;
```

b. Explain how insertions and deletions are performed on Circular Queues using Arrays? What assumptions are made for this implementation?

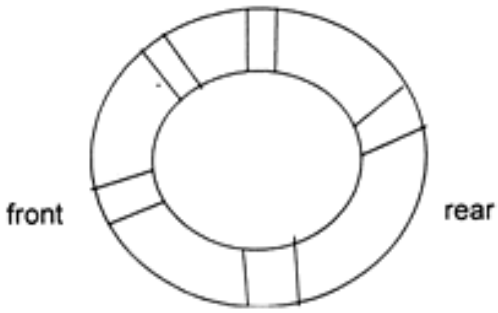
Answer:

If the queue is represented using array when the rear points to the end, insertion will be denied even if room is available at the front.One way to remove this is using a circular array.

Physically, a circular array is the same as ordinary array,say , a[n], but logically it implies that a[0] comes after a[n-1] or after a[n-1],a[0] appears.



Circular queue(physical)



The following assumptions are made for circular queue:-

1. *front* will always be pointing to the first element .front =-1 will indicate that the queue is empty.
2. Each time a new element is inserted in to the queue the rear is incremented by 1 i.e. rear=rear+1.
3. Each time an element is deleted from the queue the value of front is incremented by 1. i.e. front=front+1.
4. If front=rear then queue contains only one element (except front=rear=-1)

Q.6 a. Write a function to insert a node with data value n in a sorted linked list pointed to by p*.

Answer: Refer Page 286 of Text Book

b. Explain how a singly linked list can be used for representing a polynomial.

Answer: Refer Page 303 of Text Book

Q.7 a. By explaining the concept of deletion of a node from a double linked list, write an algorithm to delete a node from a doubly linked list.

Answer:
The deletion operation is used to delete an element from a double linked list.Before performing deletion one should remember the following points.

1. If the list is empty , deletion is not possible, output should display the message "Deletion not possible".
2. If the list contains one node, after deletion the list will be empty.
3. If the node to be deleted is not found, it should display a message " Node not found"
4. If the node is deleted , the memory space for the deleted node is deallocated.

There are various positions where a node can be deleted

a. Delete the first node.
b. Delete the last node
c. Delete the node at a specified location
d. Delete a node after a given node.

Algorithm:-

*star*t holds the address of the first node.

1. Set temp= start
2. If start==NULL then write 'UNDERFLOW' and exit.
3. Set start = next[start]
4. Set prev[start]=NULL
5. Free the space associated with temp
6. Exit.

b. Write a program for building and printing the elements of a circular linked list.

Answer: Refer Page 315 of Text Book

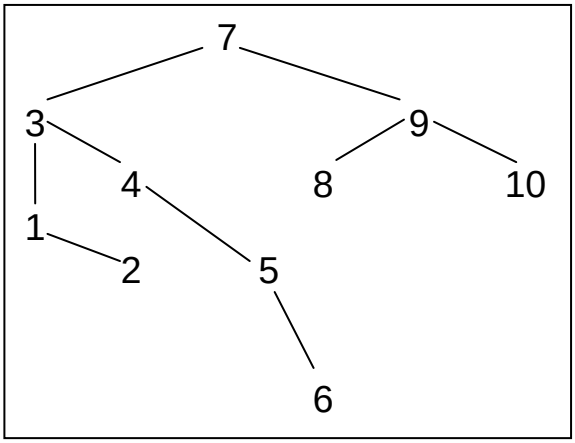
Q.8 a. Explain the formation of Binary Search Tree, by taking any suitable example.

Answer:
A binary search tree is a binary tree which is either empty or satisfies the following rules:

1. The values of the key in the left child or left subtree is less than the value of the root.

2. The value of the key in the right child or right subtree is more than equal to the value of the root.
3. All the sub-trees of the left and right children observe the two rules.Following shows a binary data structure observing the rules. The number 7 is the root node of the binary tree.It has two sub-trees, the left sub-tree with node 3 and subtree with node 9. The value of left subtree is lower subtree node is lower than the value of the root and the value of the right subtree node is higher than the vlue of the root.. This attribute is seen in all down-below nodes to the left and right of the root.

Typical diagram of a Binary Tree:-



b. Explain the formation of a binary tree from its given preorder and postorder traversal using suitable example.

Answer:

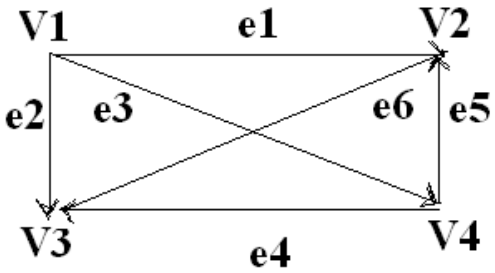
1. The first node in the preorder traversal and last node of the postorder traversal is considered as the root node of the tree.

2. Find the succesor of the root node in preorder traversal, say, n1 and predecessor of the root node in postorder traversal say n2 , then there will be two cases:
(i) if n1=n2 then this node is considered to be the left or right child of the root node, in which the construction of trees is not unique.
(ii) if n1!=0 then n1 is considered as the left child and n2 is considered as the right child of the root node.

3.Find the position of n2 and n1 in preorder and postorder traversals respectively.Now consider the two sets of preorder and postorder traversals of left and right subtrees of the root. The first set is the nodes appear after n1 and before n2 in preorder traversal and the nodes precede to node n1 in postorder traversal.The second set is the nodes appear after n2 in preorder traversal and the nodes in between n1 and n2 in postorder traversal.

4.Taking these two sets of preorder and postorder traversal for left and right subtrees , repeat the steps 2 and 3 till the entire tree is constructed.

Q.9 a. Describe Adjacency Matrix and develop the same for the following Directed Graph.



Answer:

The adjacency matrix is a matrix with one row and one column for each vertex. The values of matrix elements are either 0 or 1. The value of 1 for row i and column j implies that the edge v(i),v(j) exists.A value of 0 indicates that there is no edge between vertices v(i) and vj , Or we can say that if graph G consists of v1,v2,v3.... vn vertices then then the adjacency matrix A= [a (i,j)] of the graph G is the n x n matrix and can be defined as :

a(i,j) = 1 if v(i) is adjacent to v(j), i.e. there is an edge between v(i) and v(j)
0 if there is no edge between v(i) and v(j)

Let:-

V= { v(1),v(2),v(3),v(4) } and E={ (v1,v2),(v1,v3),(v1,v4),(v3,v4),(v4,v2) }

Then the adjacency matrix of the graph :

A =		v(1)	v(2)	v(3)	v(4)
	v(1)	0	1	1	1
	v(2)	0	0	0	0
	v(3)	0	1	0	1
	v(4)	0	1	0	0

Then it is clear that number of 1's in an adjacency matrix is equal to the number of edges in the graph.

TEXT BOOK

I. C & Data Structures, P.S. Deshpande and O.G. Kakde, Dreamtech Press, 2007