DATASTRUCTURE PROGRAMS

by

MOHAMMED FASIL V

QUESTIONS

1. Sort a given list of strings
2. Reverse a string using pointers.
3. Implement Pattern matching algorithm.
4. Search an element in the 2-dimensional array
5. Append 2 arrays
6. Merge two sorted array into one sorted array
7. Search an element in the array using iterative binary search.
8. Search an element in the array using recursive binary search.
9. Implement sparse matrix
10. Implement polynomial using arrays
11. Implement singly linked list of integers.
12. Delete a given element from a singly linked list
13. Sort a singly linked list.
14. Delete an element from a singly linked list
15. Implement a doubly linked list of integers
16. Implement a circular linked list.
17. Implement polynomial using linked list
18. Addition of 2 polynomials
19. Implement Stack using array
20. Implement Stack using linked list
21. Infix expression into its postfix expression
22. Implement Queue using array
23. Implement Queue using linked list
24. Implement a binary search tree of characters.
25. Traverse a binary search tree non recursively in preorder
26. Traverse a binary search tree non recursively in inorder
27. Traverse a binary search tree non recursively in postorder
28. Traverse a binary search tree recursively in preorder
29. Traverse a binary search tree recursively inorder
30. Traverse a binary search tree recursively postorder
31. Delete an element from a binary search tree.
32. Search an element in a binary search tree
34. Implement bubble sort
35. Implement exchange sort
36. Implement selection sort.

37. Implement insertion sort.
38. Implement quick sort.
39. Implement merge sort.
40. Implement heap sort

Question no:01

Sort a given list of strings

Source Code: .

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
void main()
{
 char str[28][50],*temp;
 int i,j,n;
 clrscr();
 printf("Enter number of names:\n");
 scanf("%d",&n);
 printf("Enter the names:\n");
 for(i=0;i<n;i++)
 {
  printf("%d:",i+1);
  gets(str[i]);
 }
 for(i=0;i<n;i++)
 {
   for(j=i+1;j<n;j++)
   if(strcmp(str[i],str[j])>0)
   {
    strcpy(temp,str[j]);
    strcpy(str[j],str[i]);
    strcpy(str[i],temp);
   }
 }
 printf("The sorted array is:\n");
 for(i=0;i<n;i++)
 printf("%d  : %s\n",i+1,str[i]);
 getch();
}
```

Question no:02

Reverse a string using pointers

Source code :

```
#include<stdio.h>
#include<conio.h>
void reverse(char *);
int length(char *);
void main()
{
 char a[100];
 clrscr();
 printf("Enter the string:\n");
 gets(a);
 reverse(a);
 printf("the reversed string is :\n%s",a);
 getch();
}
int length(char *str)
{
  int c=0;
  while(*(str+c)!='\0')
  c++;
  return c;
}
void reverse(char *s)
{
  int l,i;
  char temp,*begin,*end;
  l=length(s);
  begin=s;
  end=s;
  end=end+l-1;
  for(i=0;i<l/2;i++)
  {
   temp=*begin;
   *begin=*end;
   *end=temp;
   end--;
   begin++;
  }
}
```

Question no:03

Implement Pattern matching algorithm

Source code :

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
int match(char *,char *);
void main()
{
 char *text,*find;
 int pos;
 clrscr();
 printf("Enter the text:\n");
 gets(text);
 printf("Enter the text to find:\n");
 gets(find);
 pos=match(text,find);
 if(pos<0)
 printf("The text is not found\n");
 else
  printf("The text is found at %dth position\n",pos+1);
 getch();
}
int match(char *text,char *str)
{
  int l1,l2,i,j,len=0,index;
  l1=strlen(text);
  l2=strlen(str);
  if(l2>l1)
  index=-1;
  else
  {
   for(i=0;i<l1-l2+1;i++)
   if(text[i]==str[0])
   {
    for(j=0;j<l2;j++)
```

```
     {
     if(text[i+j]==str[j])
     len++;
     }
     if(len==l2)
     index=i;
     else
     index=-1;
    }
   }
  return index;
 }
```

Question no:04

Search an element in the 2-dimensional array

Source code :

```
#include<stdio.h>
#include<conio.h>
void main()
{
  int ar[5][5],i,j,m,n,num,flag=0,row-1,col=-1;
  clrscr();
  printf("Enter the size of matrix row X column:\n");
  scanf("%d%d",&n,&m);
  printf("Enter the elements to the matrix:\n");
  for(i=0;i<n;i++)
   for(j=0;j<m;j++)
     scanf("%d",&ar[i][j]);
  printf("Enter the element to search:\n");
  scanf("%d",&num);
  for(i=0;i<n;i++)
  {
   for(j=0;j<m;j++)
     if(ar[i][j]==num)
     {
      flag=1;
      row=i;
      col=j;
     }
   }
   if(flag==1)
   printf("the element is found at %d th row and %dth
column:\n",row+1,col+1);
   else
   printf("element is not found:\n");
  getch();
}
```

Question no:05

Append 2 arrays

Source code :

```
#include<stdio.h>
#include<conio.h>
void main()
{
  int n,ar[100],ar2[100],ar3[100],i,m;
  clrscr();
  printf("Enter the size of the fist array:\n");
  scanf("%d",&n);
  printf("Enter the elements:\n");
  for(i=0;i<n;i++)
  scanf("%d",&ar[i]);
  printf("Enter the size of the second array:\n");
  scanf("%d",&m);
  printf("Enter the elements:\n");
  for(i=0;i<m;i++)
  scanf("%d",&ar2[i]);
  for(i=0;i<n;i++)
  ar3[i]=ar[i];
  for(i=0;i<m;i++)
  ar3[i+n]=ar2[i];
  printf("the merged array is :\n");
  for(i=0;i<m+n;i++)
  printf("\nar[%d]=%d",i,ar3[i]);
  getch();
}
```

Question no:06

Merge two sorted array into one sorted array.

Source code :

```
#include<stdio.h>
#include<conio.h>
void main()
{
 int a[50],b[50],ar[100],m,n,i,j,index=0;
 clrscr();
 printf("Enter size of first matrix:\n");
 scanf("%d",&m);
 printf("Enter size of second matrix:\n");
 scanf("%d",&n);
 printf("Enter elements to first matrix:\n");
 for(i=0;i<m;i++)
 scanf("%d",&a[i]);
 printf("Enter elements to second matrix:\n");
 for(i=0;i<n;i++)
 scanf("%d",&b[i]);
 i=j=0;
 while(i<m&&j<n)
 {
  if(a[i]<=b[j])
  {
   ar[index]=a[i];
   i++;
  }
  else
  {
   ar[index]=b[j];
   j++;
  }
  index++;
 }
  if(j==n)
  {
    for(;i<m;i++,index++)
     ar[index]=a[i];
  }
  else if(i==m)
  {
   for(;j<n;j++,index++)
    ar[index]=b[j];
  }
 printf("The sorted array is :\n");
 for(i=0;i<m+n;i++)
 printf("%d\n",ar[i]);
 getch();
}
```

Question no:07

Search an element in the array using iterative binary search.

Source code :

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
int smallest(int arr[], int k, int n);
void selection_sort(int arr[], int n);
void main()
{
        int arr[100], num, i, n, beg, end, mid, found=0;
         clrscr();
        printf("\n Enter the number of elements in the array: ");
        scanf("%d", &n);
        printf("\n Enter the elements: ");
        for(i=0;i<n;i++)
                scanf("%d", &arr[i]);
        selection_sort(arr, n);
        printf("\n The sorted array is: \n");
        printf("\n\n Enter the number that has to be searched: ");
        scanf("%d", &num);
        beg = 0, end = n-1;
        while(beg<=end)
        {
                mid = (beg + end)/2;
                if (arr[mid] == num)
                {
                        printf("\n %d is present in the array at position %d",
                        num, mid+1);
                        found =1;
                        break;
                }
                if(arr[mid]<num)
                end = mid-1;
                else
                beg = mid+1;
        }
        if (beg > end && found == 0)
                printf("\n %d does not exist in the array", num);
        getch();
}
int smallest(int arr[], int k, int n)
{
        int pos = k, small=arr[k], i;
        for(i=k+1;i<n;i++)
        {
                if(arr[i]< small)
                {
                        small = arr[i];
                        pos = i;
                }
        }
        return pos;
}
void selection_sort(int arr[],int n)
{
        int k, pos, temp;
        for(k=0;k<n;k++)
        {
                pos = smallest(arr, k, n);
                temp = arr[k];
                arr[k] = arr[pos];
                arr[pos] = temp;
        }
}
```

Question no:08

Search an element in the array using recursive binary search

Source code :

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
int smallest(int arr[], int k, int n);
void selection_sort(int arr[], int n);
int binarysearch(int arr[], int beg,int end,int data);
void main()
{
	int arr[100],num,n,i,found=0;
	clrscr();
	printf("\n Enter the number of elements in the array: ");
	scanf("%d", &n);
	printf("\n Enter the elements: ");
	for(i=0;i<n;i++)
		scanf("%d", &arr[i]);
	selection_sort(arr, n);
	printf("\n\n Enter the number that has to be searched: ");
	scanf("%d", &num);
	found=binarysearch(arr,0,n-1,num);
	if(found == 0)
		printf("\n %d does not exist in the array", num);
	getch();
}
int smallest(int arr[], int k, int n)
{
	int pos = k, small=arr[k], i;
	for(i=k+1;i<n;i++)
	{
		if(arr[i]< small)
		{
			small = arr[i];
			pos = i;
		}
```

```
        }
        return pos;
}
void selection_sort(int arr[],int n)
{
        int k, pos, temp;
        for(k=0;k<n;k++)
        {
                pos = smallest(arr, k, n);
                temp = arr[k];
                arr[k] = arr[pos];
                arr[pos] = temp;
        }
}
int binarysearch(int arr[], int beg,int end,int data)
{
        int mid = (beg + end)/2;
        printf("\n%d\t%d\t",beg,end);
                if (arr[mid] == data)
                {
                        printf("\n %d is present in the array at position %d",
                        data, mid+1);
                        return 1;
                }
                if(beg>end)
                        return 0;
                if(arr[mid]>data)
                        end = mid-1;
                else
                        beg = mid+1;
                binarysearch(arr,beg,end,data);
}
```

Question no:09

Implement sparse matrix

Source code :

```
#include <stdio.h>
#include <string.h>
include<conio.h>
void main()
{
 int data,i,j,rep[3][50],m,n,index=0,choice;
 char *str[]={"Row","column","Value"};
 clrsccr()
 do
 {
  printf("Enter the order of matrix:\n");
            scanf("%d%d",&m,&n);
            printf("Enter the elements:%d%d\n",m,n);
            for(i=0;i<m;i++)
            {
              for(j=0;j<n;j++)
              {
               printf("ar[%d][%d]=",i+1,j+1);
               scanf("%d",&data);
               if(data!=0)
               {
                rep[0][index]=i;
                rep[1][index]=j;
                rep[2][index]=data;
                index++;
               }
              }
           }
    printf("The sparse matrix representation is:\n");
            for(i=0;i<3;i++)
            {
             printf("\n%s\t",str[i]);
             for(j=0;j<index;j++)
             printf("%d\t",rep[i][j]);
            }
         printf("\npress 1 to continue and 2 exit\n");
```

```
          scanf("%d",&choice);
 }
while(choice==1);
 getch();
}
```

Question no:10

Implement polynomial using arrays..

Source code :

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
struct node
{
int num;
int exp;
}a[100];
typedef struct node node;
void createpoly(node a[]);
void displaypoly(node a[]);
int n;
void main()
{
        int op;
        clrscr();
        do
        {
                printf("\n******Main Menu******");
                printf("\n1:Read polynomial\n2:Display
polynomial\n3:Exit");
                printf("\nchoice: ");
                scanf("%d",&op);
                switch(op)
                {
                        case 1:createpoly(a);break;
                        case 2:displaypoly(a);break;
                        case 3:exit(0);break;
                        default:printf("\ninvalid input");
                }
        }
```

```
	while(op!=3);
	getch();
}
void createpoly(node a[])
{
	int i;
	printf("\nEnter number of terms in the polynomial:");
	scanf("%d",&n);
	for(i=0;i<n;i++)
	{
		printf("\n%d th term:",i+1);
		printf("\ncoefficient:");
		scanf("%d",&a[i].num);
		printf("\nexponent:");
		scanf("%d",&a[i].exp);
	}
}
void displaypoly(node a[])
{
	int i;
	printf("\n");
	for(i=0;i<n;i++)
	{
		if(a[i].exp==1)
			printf("%dx+",a[i].num);
		else if(a[i].exp==0)
			printf("%d",a[i].num);
		else
			printf("%dx^(%d)+",a[i].num,a[i].exp);
	}
}
```

Question no:11

Implement singly linked list of integers..

Source code :

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<malloc.h>
struct node
{
  int data;
  struct node *next;
};
typedef struct node node;
node *start=NULL;
node *display(node *);
node *createlist(node *);
node *deletenode(node *);
node *deletevalue(node *start);
node *deletebeg(node *);
node *deletend(node *);
node *deleteafter(node *);
node *deletelist(node *);
node *insertnode(node *);
node *insertbeg(node *,node *);
node *insertafter(node *,node *);
node *insertend(node *,node *);
int check(node *,int );
void Exit(node *);
void main()
{
 int n;
 clrscr();
 do
 {
  printf("------Main Menu------\n");
  printf("1:Create list\n2:insert a node\n3 Display the list”);
  printf(“\n4:Delete a node\n5:delete list \n6:Exit\n");
  printf("choice:\0\b");
  scanf("%d",&n);
   switch(n)
   {
    case 1:start=createlist(start);
            printf("\nLinked list is created:\n");
            break;
    case 2:start=insertnode(start);break;
    case 3:start=display(start);break;
    case 4:start=deletenode(start);break;
    case 5:start=deletelist(start);
            printf("\nlist deleted succesfully\n");
            break;
    case 6:Exit(start);break;
    default:printf("invalid option:\n");
   }
 }
 while(n!=6);
 getch();
}
node *display(node *start)
{
 node *ptr;
 ptr=start;
 if(ptr==NULL)
 printf("The list is empty:\n");
 else
 {
  printf("The list is:\n");
  while(ptr!=NULL)
  {
   printf("%d\n",ptr->data);
   ptr=ptr->next;
  }
 }
 return start;
}
node *createlist(node *start)
{
 node *new_node,*ptr;
 int op,data;
 do
 {
  printf("Enter the data:\n");
  scanf("%d",&data);
  new_node=(node*)malloc(sizeof(node));
  new_node->data=data;
  if(start==NULL)
  start=insertbeg(start,new_node);
  else
  start=insertend(start,new_node);
  printf("Do you want to add a new node\n");
  printf("1:Yes\n2:No\n");
  scanf("%d",&op);
 }
 while(op==1);
 return start;
}
node *deletenode(node *start)
{
 int ch;
 do
 {
 printf("\n--------Menu---------\n");
 printf("1:delete a given value node\n2:Delete first node”);
 printf(“ \n3:Delete last node “);
 printf(“\n4:Delete next node of given value\n5:EXIT\n");
 printf("\nchoice:\0\b");
 scanf("%d",&ch);
 switch(ch)
 {
          case 1:start=deletevalue(start);break;
          case 2:start=deletebeg(start);break;
          case 3:start=deletend(start);break;
          case 4:start=deleteafter(start);break;
 }
 }while(ch!=5);
 return start;
}
node *deletevalue(node *start)
{
 node *ptr,*preptr;
 int value;
 ptr=start;
 printf("Enter the value to delete:\n");
 scanf("%d",&value);
 if(check(start,value)==1)
 {
     if(value==ptr->data)
     {
       start=deletebeg(start);
     }
     else
     {
       while(value!=ptr->data)
       {
        preptr=ptr;
        ptr=ptr->next;
       }
       if(ptr->next==NULL)
       {
         start=deletend(start);
       }
      else
      {
        preptr->next=ptr->next;
        free(ptr);
      }
     }
     printf("\nElement deleted succesfully\n");
 }
 else
 printf("\nThe given value is not matched with any node");
 return start;
}
node *deletebeg(node *start)
{
   node *ptr;
   ptr=start;
   start=ptr->next;
   free(ptr);
   printf("\nElement deleted succesfully\n");
   return start;
}
node *deletend(node *start)
{
node *ptr,*preptr;
ptr=start;
while(ptr->next!=NULL)
{
 preptr=ptr;
 ptr=ptr->next;
}
preptr->next=NULL;
free(ptr);
printf("\nElement deleted succesfully\n");
return start;
```

```
}
node *deletelist(node *start)
{
 node *ptr;
 ptr=start;
 if(start!=NULL)
 {
 while(ptr!=NULL)
 {
   start=deletebeg(start);
   ptr=start;
 }
 }
 return start;
}
void Exit(node *start)
{
 if(start!=NULL)
 {
  start=deletelist(start);
 }
 exit(0);
}
node *insertnode(node *start)
{
  int data,ch;
  node *new_node;
  do
  {
  printf("\n---------Menu-----------\n");
  printf("1:insert at begning\n2:insert after a value\n3:insert at
end\n4:Exit\n");
  printf("choice:\0\b");
  scanf("%d",&ch);
  if(ch!=4)
 {
   printf("\nEnter the value to insert:\n");
   scanf("%d",&data);
   new_node=(node*)malloc(sizeof(node));
   new_node->data=data;
   switch(ch)
   {
    case 1:start=insertbeg(start,new_node);break;
    case 2:start=insertafter(start,new_node);break;
    case 3:start=insertend(start,new_node);break;
   }
 }
}while(ch!=4);
 return start;
}
node *insertbeg(node *start,node *n)
{
 if(start==NULL)
 {
   start=n;
   start->next=NULL;
 }
 else
 {
  n->next=start;
  start=n;
 }
 printf("\nnode inserted successfully\n");
 return start;
}
node *insertafter(node *start,node *n)
{
  int val;
  node *ptr;
  ptr=start;
  printf("\nEnter the value of previouse node\n");
  scanf("%d",&val);
  if(check(start,val)==1)
  {
        while(ptr->data!=val)
        ptr=ptr->next;
        if(ptr->data==val)
        {
         n->next=ptr->next;
         ptr->next=n;
        }
        printf("\nnode inserted successfully\n");
  }
  else
  printf("\nthe given value is not matched with any node \n");
  return start;
}
node *insertend(node *start,node *n)
{
  node *ptr;
  ptr=start;
  while(ptr->next!=NULL)
  ptr=ptr->next;
  ptr->next=n;
  n->next=NULL;
  printf("\nnode inserted successfully\n");
  return start;
}
node *deleteafter(node *start)
{
         int val;
  node *ptr,*p;
  ptr=start;
  printf("\nEnter the value of previouse node\n");
  scanf("%d",&val);
  if(check(start,val)==1)
  {
     while(ptr->data!=val)
     ptr=ptr->next;
     if(ptr->data==val)
    {
     p=ptr->next;
     p=p->next;
     free(ptr->next);
     ptr->next=p;
    }
    printf("\nElement deleted succesfully\n");
  }
  else
  printf("\nthe given value is not matched with any node \n");
  return start;
}
int check(node *start,int value)
{
        node *ptr;
        int bool=0;
        for(ptr=start;ptr!=NULL;ptr=ptr->next)
        if(ptr->data==value)
        bool=1;
        return bool;
}
```

Question no:12

Delete a given element from a singly linked list.

Source code :

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<malloc.h>
struct node
{
  int data;
  struct node *next;
};
typedef struct node node;
node *start=NULL;
node *display(node *);
node *createlist(node *);
node *deletenode(node *);
node *deletebeg(node *);
node *deletend(node *);
node *deletelist(node *);
int check(node *start,int value)
{
        node *ptr;
        int bool=0;
        for(ptr=start;ptr!=NULL;ptr=ptr->next)
        if(ptr->data==value)
        bool=1;
        return bool;
}
void main()
{
 int n;
 clrscr();
 do
 {
  printf("------Main Menu------\n");
  printf("1:Create a node\n2:Display the list\n3:Delete a
node\n4:delete list \n5:Exit\n");
  printf("choice:\0\b");
  scanf("%d",&n);
   switch(n)
   {
    case 1:start=createlist(start);
            printf("\nLinked list is created:\n");
            break;
    case 2:start=display(start);
            break;
    case 3:start=deletenode(start);break;
    case 4:start=deletelist(start);
            printf("\nlist deleted succesfully\n");
            break;
    case 5:break;
    default:printf("invalid option:\n");
   }
 }
 while(n!=5);
 getch();
}
node *display(node *start)
{
 node *ptr;
 ptr=start;
 if(ptr==NULL)
 printf("The list is empty:\n");
 else
 {
  printf("The list is:\n");
  while(ptr!=NULL)
  {
   printf("%d\n",ptr->data);
   ptr=ptr->next;
  }
 }
 return start;
}
node *createlist(node *start)
{
 node *new_node,*ptr;
 int op,data;
 do
 {
  printf("Enter the data:\n");
  scanf("%d",&data);
  new_node=(node*)malloc(sizeof(node));
  new_node->data=data;
  if(start==NULL)
  {
   start=new_node;
   new_node->next=NULL;
  }
  else
  {
    ptr=start;
    while(ptr->next!=NULL)
    ptr=ptr->next;
    ptr->next=new_node;
    new_node->next=NULL;
  }
  printf("Do you want to add a new node\n");
  printf("1:Yes\n2:No\n");
  scanf("%d",&op);
 }
 while(op==1);
 return start;
}
node *deletenode(node *start)
{
node *ptr,*preptr;
 int value;
 ptr=start;
 printf("Enter the value to delete:\n");
 scanf("%d",&value);
 if(check(start,value)==1)
 {
     if(value==ptr->data)
     {
       start=deletebeg(start);
     }
     else
     {
       while(value!=ptr->data)
       {
        preptr=ptr;
        ptr=ptr->next;
       }
       if(ptr->next==NULL)
       {
         start=deletend(start);
       }
      else
      {
        preptr->next=ptr->next;
        free(ptr);
      }
     }
     printf("\nElement deleted succesfully\n");
 }
 else
 printf("\nThe given value is not matched with any node");
 return start;
}
node *deletebeg(node *start)
{
node *ptr;
ptr=start;
start=ptr->next;
free(ptr);
return start;
}
node *deletend(node *start)
{
node *ptr,*preptr;
ptr=start;
while(ptr->next!=NULL)
{
 preptr=ptr;
 ptr=ptr->next;
}
preptr->next=NULL;
free(ptr);
return start;
}
node *deletelist(node *start)
{
 node *ptr;
 ptr=start;
 if(start!=NULL)
 {
 while(ptr!=NULL)
 {
   start=deletebeg(start);
   ptr=start;
 }
 }
 return start;
}
```

Question no:13

Sort a singly linked list.

Source code :

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<malloc.h>
struct node
{
  int data;
  struct node *next;
};
typedef struct node node;
node *start=NULL;
node *display(node *);
node *createlist(node *);
node *deletelist(node *);
node *sortlist(node *);
node *deletebeg(node *);
void main()
{
 int n;
 clrscr();
 do
 {
  printf("------Main Menu------\n");
  printf("1:Create a node\n2:Display the list\n3:Delete the list”);
  printf(“\n4:sort list\n5:Exit\n");
  printf("choice:\0\b");
  scanf("%d",&n);
   switch(n)
   {
    case 1:start=createlist(start);
            printf("\nLinked list is created:\n");
            break;
    case 2:start=display(start);
```

```
            break;
    case 3:start=deletelist(start);
            printf("list deleted succesfully\n");
            break;
    case 4:start=sortlist(start);break;
    case 5:break;
    default:printf("invalid option:\n");
   }
 }
 while(n!=5);
 getch();
}
node *display(node *start)
{
 node *ptr;
 ptr=start;
 if(ptr==NULL)
 printf("The list is empty:\n");
 else
 {
  printf("The list is:\n");
  while(ptr!=NULL)
  {
   printf("%d\n",ptr->data);
   ptr=ptr->next;
  }
 }
 return start;
}
node *createlist(node *start)
{
 node *new_node,*ptr;
 int op,data;
 do
 {
  printf("Enter the data:\n");
  scanf("%d",&data);
  new_node=(node*)malloc(sizeof(node));
  new_node->data=data;
  if(start==NULL)
  {
   start=new_node;
   new_node->next=NULL;
  }
  else
  {
    ptr=start;
    while(ptr->next!=NULL)
    ptr=ptr->next;
    ptr->next=new_node;
    new_node->next=NULL;
  }
  printf("Do you want to add a new node\n");
  printf("1:Yes\n2:No\n");
  scanf("%d",&op);
 }
 while(op==1);
 return start;
}
node *deletelist(node *start)
{
 node *ptr;
 ptr=start;
 if(start!=NULL)
 {
 while(ptr!=NULL)
 {
   start=deletebeg(start);
   ptr=start;
 }
 }
 return start;
}
node *deletebeg(node *start)
```

```
{
 node *ptr;
 ptr=start;
 start=ptr->next;
 free(ptr);
 return start;
}
node *sortlist(node *start)
{
  node *ptr1,*ptr2;
  int temp;
  ptr1=start;
  while(ptr1!=NULL)
  {
    ptr2=ptr1->next;
    while(ptr2!=NULL)
    {
     if(ptr1->data>ptr2->data)
     {
      temp=ptr1->data;
      ptr1->data=ptr2->data;
      ptr2->data=temp;
     }
     ptr2=ptr2->next;
    }
    ptr1=ptr1->next;
  }
  return start;
}
```

Question no:14

Delete an element from a singly linked list

Source code :

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <malloc.h>
struct node
{
  int data;
  struct node *next;
};
typedef struct node node;
node *start=NULL;
node *display(node *);
node *createlist(node *);
node *deletenode(node *);
node *deletevalue(node *start);
node *deletebeg(node *);
node *deletend(node *);
node *deleteafter(node *);
node *deletelist(node *);
node *insertbeg(node *,node *);
node *insertend(node *,node *);
int check(node *,int );
void Exit(node *);
void main()
{
 int n;
 clrscr();
 do
 {
  printf("------Main Menu------\n");
  printf("1:Create list\n2: Display the list\n3:Delete a node\n4:delete
list \n5:Exit\n");
  printf("choice:\0\b");
  scanf("%d",&n);
   switch(n)
   {
    case 1:start=createlist(start);
            printf("\nLinked list is created:\n");
            break;
          case 2:start=display(start);break;
    case 3:start=deletenode(start);break;
    case 4:start=deletelist(start);
            printf("\nlist deleted succesfully\n");
            break;
    case 5:Exit(start);break;
    default:printf("invalid option:\n");
   }
 }
 while(n!=6);
 getch();
}
node *display(node *start)
{
 node *ptr;
 ptr=start;
 if(ptr==NULL)
 printf("The list is empty:\n");
 else
 {
  printf("The list is:\n");
  while(ptr!=NULL)
  {
   printf("%d\n",ptr->data);
   ptr=ptr->next;
  }
 }
 return start;
}
node *createlist(node *start)
{
 node *new_node,*ptr;
 int op,data;
 do
 {
  printf("Enter the data:\n");
  scanf("%d",&data);
  new_node=(node*)malloc(sizeof(node));
  new_node->data=data;
  if(start==NULL)
  start=insertbeg(start,new_node);
  else
  start=insertend(start,new_node);
  printf("Do you want to add a new node\n");
  printf("1:Yes\n2:No\n");
  scanf("%d",&op);
 }
 while(op==1);
 return start;
}
node *deletenode(node *start)
{
 int ch;
 do
 {
 printf("\n--------Menu---------\n");
 printf("1:delete a given value node\n2:Delete first node \n3:Delete
last node \n4:Delete next node of given value\n5:EXIT\n");
 printf("\nchoice:\0\b");
 scanf("%d",&ch);
 switch(ch)
 {
          case 1:start=deletevalue(start);break;
          case 2:start=deletebeg(start);break;
          case 3:start=deletend(start);break;
          case 4:start=deleteafter(start);break;
 }
 }while(ch!=5);
 return start;
}
node *deletevalue(node *start)
{
 node *ptr,*preptr;
 int value;
 ptr=start;
 printf("Enter the value to delete:\n");
 scanf("%d",&value);
 if(check(start,value)==1)
 {
     if(value==ptr->data)
     {
       start=deletebeg(start);
     }
     else
     {
       while(value!=ptr->data)
       {
        preptr=ptr;
        ptr=ptr->next;
       }
       if(ptr->next==NULL)
       {
         start=deletend(start);
       }
      else
      {
        preptr->next=ptr->next;
        free(ptr);
      }
     }
     printf("\nElement deleted succesfully\n");
 }
 else
 printf("\nThe given value is not matched with any node");
 return start;
}
node *deletebeg(node *start)
{
   node *ptr;
   ptr=start;
   start=ptr->next;
   free(ptr);
   printf("\nElement deleted succesfully\n");
   return start;
}
node *deletend(node *start)
{
node *ptr,*preptr;
ptr=start;
while(ptr->next!=NULL)
{
 preptr=ptr;
 ptr=ptr->next;
}
preptr->next=NULL;
free(ptr);
printf("\nElement deleted succesfully\n");
return start;
}
node *deletelist(node *start)
{
 node *ptr;
 ptr=start;
 if(start!=NULL)
 {
 while(ptr!=NULL)
 {
   start=deletebeg(start);
   ptr=start;
 }
 }
 return start;
}
void Exit(node *start)
{
 if(start!=NULL)
 {
  start=deletelist(start);
 }
 exit(0);
}
node *insertbeg(node *start,node *n)
{
 if(start==NULL)
 {
   start=n;
   start->next=NULL;
 }
 else
 {
  n->next=start;
  start=n;
 }
 printf("\nnode inserted successfully\n");
 return start;
}
```

```
node *insertend(node *start,node *n)
{
  node *ptr;
  ptr=start;
  while(ptr->next!=NULL)
  ptr=ptr->next;
  ptr->next=n;
  n->next=NULL;
  printf("\nnode inserted successfully\n");
  return start;
}
node *deleteafter(node *start)
{
        int val;
  node *ptr,*p;
  ptr=start;
  printf("\nEnter the value of previouse node\n");
  scanf("%d",&val);
  if(check(start,val)==1)
  {
     while(ptr->data!=val)
     ptr=ptr->next;
     if(ptr->data==val)
    {
     p=ptr->next;
     p=p->next;
     free(ptr->next);
     ptr->next=p;
    }
    printf("\nElement deleted succesfully\n");
  }
  else
  printf("\nthe given value is not matched with any node \n");
  return start;
}
```

```
int check(node *start,int value)
{
        node *ptr;
        int bool=0;
        for(ptr=start;ptr!=NULL;ptr=ptr->next)
        if(ptr->data==value)
        bool=1;
        return bool;
}
```

Question no:15

Implement a doubly linked list of integers

Source code :

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <malloc.h>
struct node
{
  struct node *prev;
  int data;
  struct node *next;
};
typedef struct node node;
node *start=NULL;
node *display(node *);
node *createlist(node *);
node *deletenode(node *);
node *deletevalue(node *start);
node *deletebeg(node *);
node *deletend(node *);
node *deleteafter(node *);
node *deletelist(node *);
node *insertnode(node *);
node *insertbeg(node *,node *);
node *insertafter(node *,node *);
node *insertend(node *,node *);
int check(node *,int );
void Exit(node *);
void main()
{
 int n;
 clrscr();
 do
 {
  printf("------Main Menu------\n");
  printf("1:Create list\n2:insert a node\n3 Display the list\n4:Delete a
node\n5:delete list \n6:Exit\n");
  printf("choice:\0\b");
  scanf("%d",&n);
   switch(n)
   {
    case 1:start=createlist(start);
             printf("\nLinked list is created:\n");
             break;
    case 2:start=insertnode(start);break;
    case 3:start=display(start);break;
    case 4:start=deletenode(start);break;
    case 5:start=deletelist(start);
             printf("\nlist deleted succesfully\n");
             break;
    case 6:Exit(start);break;
    default:printf("invalid option:\n");
   }
 }
 while(n!=6);
 getch();
}
node *display(node *start)
{
 node *ptr;
 ptr=start;
 if(ptr==NULL)
 printf("The list is empty:\n");
 else
 {
  printf("The list is:\n");
  while(ptr!=NULL)
  {
   printf("%d\n",ptr->data);
   ptr=ptr->next;
  }
 }
 return start;
}
node *createlist(node *start)
{
 node *new_node;
 int op,data;
 do
 {
  printf("Enter the data:\n");
  scanf("%d",&data);
  new_node=(node*)malloc(sizeof(node));
  new_node->data=data;
  if(start==NULL)
  start=insertbeg(start,new_node);
  else
  start=insertend(start,new_node);
  printf("Do you want to add a new node\n");
  printf("1:Yes\n2:No\n");
  scanf("%d",&op);
 }
 while(op==1);
 return start;
}
node *deletenode(node *start)
{
 int ch;
 do
 {
 printf("\n--------Menu---------\n");
 printf("1:delete a given value node\n2:Delete first node”);
 printf(“ \n3:Delete last node \n”);
 printf(“4:Delete next node of given value\n5:EXIT\n");
 printf("\nchoice:\0\b");
 scanf("%d",&ch);
 switch(ch)
 {
         case 1:start=deletevalue(start);break;
         case 2:start=deletebeg(start);break;
         case 3:start=deletend(start);break;
         case 4:start=deleteafter(start);break;
 }
 }while(ch!=5);
 return start;
}
node *deletevalue(node *start)
{
 node *ptr,*preptr;
 int value;
 ptr=start;
 printf("Enter the value to delete:\n");
 scanf("%d",&value);
 if(check(start,value)==1)
 {
    if(value==ptr->data)
    {
      start=deletebeg(start);
    }
    else
    {
      while(value!=ptr->data)
      {
       preptr=ptr;
       ptr=ptr->next;
      }
      if(ptr->next==NULL)
      {
       start=deletend(start);
      }
     else
     {
       preptr->next=ptr->next;
       ptr->next->prev=preptr;
       free(ptr);
     }
    }
    printf("\nElement deleted succesfully\n");
 }
 else
 printf("\nThe given value is not matched with any node");
 return start;
}
node *deletebeg(node *start)
{
   node *ptr;
   ptr=start;
   start=ptr->next;
   start->prev=NULL;
   free(ptr);
   printf("\nElement deleted succesfully\n");
   return start;
}
node *deletend(node *start)
{
  node *ptr,*preptr;
  ptr=start;
  while(ptr->next!=NULL)
  {
   preptr=ptr;
   ptr=ptr->next;
  }
  preptr->next=NULL;
  free(ptr);
  printf("\nElement deleted succesfully\n");
  return start;
}
node *deletelist(node *start)
{
 node *ptr;
 ptr=start;
 if(start!=NULL)
 {
 while(ptr!=NULL)
 {
   start=deletebeg(start);
   ptr=start;
 }
 }
 return start;
}
void Exit(node *start)
{
 if(start!=NULL)
 {
  start=deletelist(start);
 }
 exit(0);
}
node *insertnode(node *start)
{
  int data,ch;
  node *new_node;
  do
  {
  printf("\n---------Menu-----------\n");
  printf("1:insert at begning\n2:insert after a value\n3:insert at
end\n4:Exit\n");
```

```
  printf("choice:\0\b");
  scanf("%d",&ch);
  if(ch!=4)
  {
    printf("\nEnter the value to insert:\n");
    scanf("%d",&data);
    new_node=(node*)malloc(sizeof(node));
    new_node->data=data;
    switch(ch)
    {
     case 1:start=insertbeg(start,new_node);break;
     case 2:start=insertafter(start,new_node);break;
     case 3:start=insertend(start,new_node);break;
    }
  }
}while(ch!=4);
  return start;
}
node *insertbeg(node *start,node *n)
{
 if(start==NULL)
 {
   start=n;
   start->next=NULL;
   start->prev=NULL;
 }
 else
 {
  start->prev=n;
  n->next=start;
  start=n;
  start->prev=NULL;
 }
 printf("\nnode inserted successfully\n");
 return start;
```

```
}
node *insertafter(node *start,node *n)
{
  int val;
  node *ptr;
  ptr=start;
  printf("\nEnter the value of previouse node\n");
  scanf("%d",&val);
  if(check(start,val)==1)
  {
        while(ptr->data!=val)
        ptr=ptr->next;
        if(ptr->data==val)
        {
         n->next=ptr->next;
         n->prev=ptr;
         ptr->next->prev=n;
         ptr->next=n;
        }
        printf("\nnode inserted successfully\n");
  }
  else
  printf("\nthe given value is not matched with any node \n");
  return start;
}
node *insertend(node *start,node *n)
{
  node *ptr;
  ptr=start;
  while(ptr->next!=NULL)
  ptr=ptr->next;
  ptr->next=n;
  n->next=NULL;
  n->prev=ptr;
  printf("\nnode inserted successfully\n");
  return start;
}
node *deleteafter(node *start)
{
        int val;
  node *ptr,*p;
  ptr=start;
  printf("\nEnter the value of previouse node\n");
  scanf("%d",&val);
  if(check(start,val)==1)
  {
    while(ptr->data!=val)
    ptr=ptr->next;
    if(ptr->data==val)
   {
    p=ptr->next;
    p=p->next;
    free(ptr->next);
    ptr->next=p;
    if(p!=NULL)
    p->prev=ptr;
   }
   printf("\nElement deleted succesfully\n");
  }
  else
  printf("\nthe given value is not matched with any node \n");
  return start;
}
int check(node *start,int value)
{
        node *ptr;
        int bool=0;
        for(ptr=start;ptr!=NULL;ptr=ptr->next)
        if(ptr->data==value)
        bool=1;
        return bool;
    }
```

Question no:16
Implement circular linked list

Source code :
```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <malloc.h>
struct node
{
  int data;
  struct node *next;
};
typedef struct node node;
node *start=NULL;
node *display(node *);
node *createlist(node *);
node *deletenode(node *);
node *deletevalue(node *start);
node *deletebeg(node *);
node *deletend(node *);
node *deleteafter(node *);
node *deletelist(node *);
node *insertnode(node *);
node *insertbeg(node *,node *);
node *insertafter(node *,node *);
node *insertend(node *,node *);
int check(node *,int );
void Exit(node *);
void main()
{
 int n;
 clrscr();
 do
 {
  printf("------Main Menu------\n");
  printf("1:Create list\n2:insert a node\n3 Display the list\n4:Delete a
node\n5:delete list \n6:Exit\n");
  printf("choice:\0\b");
  scanf("%d",&n);
   switch(n)
   {
    case 1:start=createlist(start);
           printf("\nLinked list is created:\n");
           break;
    case 2:start=insertnode(start);break;
    case 3:start=display(start);break;
    case 4:start=deletenode(start);break;
    case 5:start=deletelist(start);
           printf("\nlist deleted succesfully\n");
           break;
    case 6:Exit(start);break;
    default:printf("invalid option:\n");
   }
 }
 while(n!=6);
 getch();
 }
node *display(node *start)
{
 node *ptr;
 ptr=start;
 if(ptr==NULL)
 printf("The list is empty:\n");
 else
 {
  printf("The list is:\n");
  do
  {
   printf("%d\n",ptr->data);
   ptr=ptr->next;
  }
  while(ptr!=start);
 }
 return start;
}
node *createlist(node *start)
{
 node *new_node;
 int op,data;
 do
 {
  printf("Enter the data:\n");
  scanf("%d",&data);
  new_node=(node*)malloc(sizeof(node));
  if(new_node==NULL)
  printf("hello");
  new_node->data=data;
  if(start==NULL)
  start=insertbeg(start,new_node);
  else
  start=insertend(start,new_node);
  printf("Do you want to add a new node\n");
  printf("1:Yes\n2:No\n");
  scanf("%d",&op);
 }
 while(op==1);
 return start;
}
node *deletenode(node *start)
{
 int ch;
 do
 {
 printf("\n--------Menu---------\n");
 printf("1:delete a given value node\n2:Delete first node \n3:Delete
last node \n4:Delete next node of given value\n5:EXIT\n");
 printf("\nchoice:\0\b");
 scanf("%d",&ch);
 switch(ch)
 {
          case 1:start=deletevalue(start);break;
          case 2:start=deletebeg(start);break;
          case 3:start=deletend(start);break;
          case 4:start=deleteafter(start);break;
 }
 }while(ch!=5);
 return start;
}
node *deletevalue(node *start)
{
 node *ptr,*preptr;
 int value;
 ptr=start;
 printf("Enter the value to delete:\n");
 scanf("%d",&value);
 if(check(start,value)==1)
 {
     if(value==ptr->data)
     {
       start=deletebeg(start);
     }
     else
     {
       while(value!=ptr->data)
       {
        preptr=ptr;
        ptr=ptr->next;
       }
       if(ptr->next==NULL)
       {
         start=deletend(start);
       }
      else
      {
        preptr->next=ptr->next;
        free(ptr);
      }
     }
     printf("\nElement deleted succesfully\n");
 }
 else
 printf("\nThe given value is not matched with any node");
 return start;
}
node *deletebeg(node *start)
{
   node *ptr;
   ptr=start;
   start=ptr->next;
   free(ptr);
   printf("\nElement deleted succesfully\n");
   return start;
}
node *deletend(node *start)
{
node *ptr,*preptr;
ptr=start;
while(ptr->next!=NULL)
{
 preptr=ptr;
 ptr=ptr->next;
}
preptr->next=start;
free(ptr);
```

```
printf("\nElement deleted succesfully\n");
return start;
}
node *deletelist(node *start)
{
 node *ptr;
 ptr=start;
 if(start!=NULL)
 {
 do
 {
   start=deletebeg(start);
   ptr=start;
 }
 while(ptr!=start);
 }
 return start;
}
void Exit(node *start)
{
 if(start!=NULL)
 {
  start=deletelist(start);
 }
 exit(0);
}
node *insertnode(node *start)
{
  int data,ch;
  node *new_node;
  do
  {
  printf("\n---------Menu-----------\n");
  printf("1:insert at begning\n2:insert after a value\n3:insert at
end\n4:Exit\n");
```

```
  printf("choice:\0\b");
  scanf("%d",&ch);
  if(ch!=4)
  {
    printf("\nEnter the value to insert:\n");
    scanf("%d",&data);
    new_node=(node*)malloc(sizeof(node));
    new_node->data=data;
    switch(ch)
    {
     case 1:start=insertbeg(start,new_node);break;
     case 2:start=insertafter(start,new_node);break;
     case 3:start=insertend(start,new_node);break;
    }
  }
}while(ch!=4);
  return start;
}
node *insertbeg(node *start,node *n)
{
 if(start==NULL)
 {
   start=n;
   start->next=NULL;
 }
 else
 {
  n->next=start;
  start=n;
 }
 printf("\nnode inserted successfully\n");
 return start;
}
node *insertafter(node *start,node *n)
{
  int val;
  node *ptr;
  ptr=start;
  printf("\nEnter the value of previouse node\n");
  scanf("%d",&val);
  if(check(start,val)==1)
  {
        while(ptr->data!=val)
        ptr=ptr->next;
        if(ptr->data==val)
        {
         n->next=ptr->next;
         ptr->next=n;
        }
        printf("\nnode inserted successfully\n");
  }
  else
  printf("\nthe given value is not matched with any node \n");
  return start;
}
node *insertend(node *start,node *n)
{
  node *ptr;
  ptr=start;
  while(ptr->next!=NULL)
  ptr=ptr->next;
  ptr->next=n;
  n->next=start;
  printf("\nnode inserted successfully\n");
  return start;
}
node *deleteafter(node *start)
{
         int val;
  node *ptr,*p;
  ptr=start;
  printf("\nEnter the value of previouse node\n");
  scanf("%d",&val);
  if(check(start,val)==1)
  {
     while(ptr->data!=val)
     ptr=ptr->next;
     if(ptr->data==val)
    {
     p=ptr->next;
     p=p->next;
     free(ptr->next);
     ptr->next=p;
    }
    printf("\nElement deleted succesfully\n");
  }
  else
  printf("\nthe given value is not matched with any node \n");
  return start;
}
int check(node *start,int value)
{
          node *ptr;
          int bool=0;
          for(ptr=start;ptr!=NULL;ptr=ptr->next)
          if(ptr->data==value)
          bool=1;
          return bool;
}
```

Question no:17

Implement polynomial using linked list

Source code :

```
#include <stdio.h>
#include <conio.h>
#include <malloc.h>
#include <stdlib.h>
struct node
{
int num;
int exp;
struct node *next;
};
typedef struct node node;
node *start=NULL;
node *createpoly(node *);
node *displaypoly(node *);
void main()
{
	int op;
	clrscr();
	do
	{
		printf("\n******Main Menu******");
		printf("\n1:Read polynomial”);
		printf(“\n2:Display polynomial\n3:Exit");
		printf("\nchoice: ");
		scanf("%d",&op);
		switch(op)
		{
			case 1:start=createpoly(start);break;
			case 2:start=displaypoly(start);break;
			case 3:exit(0);break;
			default:printf("\ninvalid input");
```

```
                }
        }
        while(op!=3);
        getch();
}
node *createpoly(node *start)
{
        node *new_node,*ptr;
        int op,data,coeff;
        do
        {
                printf("Enter the number:\n");
                scanf("%d",&data);
                printf("\nEnter its coefficient:\n");
                scanf("%d",&coeff);
                new_node=(node*)malloc(sizeof(node));
                new_node->num=data;
                new_node->exp=coeff;
                new_node->next=NULL;
                if(start==NULL)
                        start=new_node;
                else
                        {
                                ptr=start;
                                while(ptr->next!=NULL)
                                        ptr=ptr->next;
                                ptr->next=new_node;
                        }
                printf("Do you want to add a new term\n");
                printf("1:Yes\n2:No\n");
                scanf("%d",&op);
        }
        while(op==1&&coeff!=0);
        return start;
}
```

```
node *displaypoly(node *start)
{
        node *ptr;
        ptr=start;
        printf("\n");
        while(ptr!=NULL)
        {
                if(ptr -> exp==1)
                        printf("%dx+", ptr -> num);
                else if(ptr -> exp==0)
                        printf("%d", ptr -> num);
                else
                        printf("%dx^(%d)+", ptr -> num, ptr ->exp);
                ptr = ptr -> next;
        }
        return start;
}
```

Question no:18

Addition of 2 polynomials

Source code :

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
struct node
{
int num;
int exp;
}a[100],b[100],c[100];
void displaypoly(struct node f[],int k);
void main()
{
        int op,i,n,j,m,index=0;
        clrscr();
        do
        {
                printf("\n******Main Menu******");
                printf("\n1:Addition\n2:Exit");
                printf("\nchoice: ");
                scanf("%d",&op);
                switch(op)
                {
                        case 1:
                                {
                                        printf("\nEnter number of terms in the 1st
polynomial:");
                                        scanf("%d",&n);
                                        printf("\nEnter 1st polynomial:");
                                        for(i=0;i<n;i++)
                                        {
                                                printf("\n%d th term:",i+1);
                                                printf("\ncoefficient:");
                                                scanf("%d",&a[i].num);
                                                printf("\nexponent:");
                                                scanf("%d",&a[i].exp);
                                        }
                                        printf("\nEnter number of terms in the
2nd polynomial:");
                                        scanf("%d",&m);
                                        printf("\nEnter 2nd polynomial:");
                                        for(i=0;i<m;i++)
                                        {
                                                printf("\n%d th term:",i+1);
                                                printf("\ncoefficient:");
                                                scanf("%d",&b[i].num);
                                                printf("\nexponent:");
                                                scanf("%d",&b[i].exp);
                                        }
                                        i=j=0;
                                        while(i<n&&j<m)
                                        {
                                                if(a[i].exp==b[j].exp)
                                                {

        c[index].num=a[i].num+b[j].num;
                                                c[index].exp=a[i].exp;
                                                i++;
                                                j++;
                                                }
                                                else if(a[i].exp<b[j].exp)
                                                {
                                                c[index].num=b[j].num;
                                                c[index].exp=b[j].exp;
                                                j++;
                                                }
                                                else if(a[i].exp>b[j].exp)
                                                {
                                                c[index].num=a[i].num;
                                                c[index].exp=a[i].exp;
                                                i++;
                                                }
```

```
                        index++;
                    }
                    if(i>n-1)
                    {
                        while(j<m)
                        {
                            c[index].num=b[j].num;
                            c[index].exp=b[j].exp;
                            j++;
                            index++;
                        }
                    }
                    else if(j>m-1)
                    {
                        while(i<n)
                        {
                            c[index].num=a[i].num;
                            c[index].exp=a[i].exp;
                            i++;
                            index++;
                        }
                    }
                    printf("\nresult is:\n");
                    displaypoly(a,n);
                    printf("+");
                    displaypoly(b,m);
                    printf("=");
                    displaypoly(c,index);
                }
            case 6:exit(0);break;
            default:printf("\ninvalid input");
        }
    }
    while(op!=2);
    getch();
}
```

```
void displaypoly(struct node f[],int k)
{
        int i;
        for(i=0;i<k;i++)
        {
                if(f[i].exp==1)
                        printf("%dx+",f[i].num);
                else if(f[i].exp==0||i==k-1)
                        printf("%d",f[i].num);
                else
                        printf("%dx^(%d)+",f[i].num,f[i].exp);
        }
}
```

Question no:19

Implement Stack using array.

Source code :

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
#define MAX 5
int st[MAX], top=-1;
int push(int st[], int top);
int pop(int st[],int top);
void peek(int st[],int top);
void display(int st[],int top);
void main()
{
int option;
clrscr();
do
{
printf("\n *****MAIN MENU*****");
printf("\n 1. PUSH");
printf("\n 2. POP");
printf("\n 3. PEEK");
printf("\n 4. DISPLAY");
printf("\n 5. EXIT");
printf("\n Enter your option: ");
scanf("%d", &option);
switch(option)
{
case 1:top=push(st,top);break;
case 2:top=pop(st,top);break;
case 3:top=peek(st,top);break;
case 4:top=display(st,top);
case 5:exit(0);
break;
}
}while(option!=5);
getch();
}
```

```
int push(int st[], int top)
{
        int val;
if(top == MAX-1)
{
printf("\n STACK OVERFLOW");
}
else
{
        printf("\nEnter the value to push to stack:\n");
        scanf("%d",&val);
top++;
st[top] = val;
printf("\n%d is pushed to stack",val);
}
return top;
}
int pop(int st[],int top)
{
int val;
if(top == -1)
{
printf("\n STACK UNDERFLOW");
return -1;
}
else
{
val = st[top];
top--;
printf("%d is poped from stack\n",val);
return top;
}
}
void display(int st[])
{
int i;
if(top == -1)
printf("\n STACK IS EMPTY");
else
{
```

```
for(i=top;i>=0;i--)
printf("\n %d",st[i]);
printf("\n");
}
}
int peek(int st[])
{
if(top == -1)
{
printf("\n STACK IS EMPTY");
return  -1;
}
else
return (st[top]);
}
```

Question no:20

Implement Stack using linked list.

Source code :

```
#include<stdio.h>
#include<malloc.h>
#include<stdlib.h>
#include<conio.h>
struct stack
{
        struct stack *prev;
        int data;
};
typedef struct stack stack;
stack *top=NULL;
stack *push(stack *);
stack *pop(stack *);
stack *peek(stack *);
stack *display(stack *);
void Exit(stack *);
void main()
{
        int op;
        clrscr();
        do
    {
      printf("\n *****MAIN MENU*****");
      printf("\n 1. PUSH");
      printf("\n 2. POP");
      printf("\n 3. PEEK");
      printf("\n 4. DISPLAY");
      printf("\n 5. EXIT");
      printf("\n choice:_\b ");
      scanf("%d", &op);
      switch(op)
      {
        case 1:top=push(top);break;
        case 2:top=pop(top);break;
        case 3:top=peek(top);break;
```

```
         case 4:top=display(top);break;
         case 5:Exit(top);break;
         default:printf("\ninvalid input\n");
        }
      }
      while(op!=5);
      getch();
  }
  stack *push(stack *top)
  {
          int val;
          stack *new_node;
          new_node=(stack*)malloc(sizeof(stack));
          printf("\nEnter the value to push to stack:\n");
          scanf("%d",&val);
          new_node->data=val;
          if(top==NULL)
          {
                  top=new_node;
                  top->prev=NULL;
          }
          else
          {
                  new_node->prev=top;
                  top=new_node;
          }
          printf("\n%d is pushed to stack succesfully",top->data);
          return top;
  }
  stack *pop(stack *top)
  {
          stack *ptr;
          if(top==NULL)
              printf("\nstack underflow\n");
          else
          {
                  printf("\n%d is poped from stack succesfully\n",top->data);
                  ptr=top;
                  top=top->prev;
                  free(ptr);
          }
          return top;
  }
  stack *peek(stack *top)
  {
          if(top==NULL)
              printf("\nstack is empty\n");
          else
                  printf("\ntop element in stack=%d\n",top->data);
          return top;
  }
  stack *display(stack *top)
  {
          stack *ptr;
          ptr=top;
          if(top==NULL)
              printf("\nstack is empty\n");
          else
          {
                  printf("\nthe elements in stack is:\n");
                  while(ptr!=NULL)
                  {
                          printf("%d\n",ptr->data);
                          ptr=ptr->prev;
                  }
          }
          return top;
  }
  void Exit(stack *top)
  {
          if(top!=NULL)
          {
                  while(top!=NULL)
                  top=pop(top);
          }
          exit(0);
  }
```

Question no:21

Infix expression into its postfix expression.

Source code :

```
#include<stdio.h>
#include<conio.h>
char stack[100];
int top=-1;
void push(char x)
{
        stack[++top]=x;
}
char pop()
{
        if(top==-1)
                return -1;
        else
                return stack[top--];
}
int priority(char x)
{
        int val;
        if(x=='(')
                val=0;
        if(x=='+'||x=='-')
                val=1;
        if(x=='*'||x=='/'||x=='%')
                val=2;
        if(x=='$'||x=='^')
                val=3;
        return val;
}
void main()
{
        char exp[100],*c,x;
        clrscr();
        printf("\nEnter the Expression:\n");
        scanf("%s",exp);
        c=exp;
        while(*c!='\0')
        {
                if(isalnum(*c)
                  printf("%c",*c);
                else if(*c=='(')
                        push(*c);
                else if(*c==')')
                {
                        while((x=pop())!='(')
                                printf("%c",x);
                }
                else
                {
                        while(priority(stack[top]>=priority(*c))
                                printf("%c",pop());
                        push(*c);
                }
                c++;
        }
        while(top!=-1)
                printf("%c",pop());
        getch();
}
```

Question no:22

Implement Queue using array.

Source code :

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
#define MAX 5
void insert();
void delete();
void display();
int queue[MAX];
int front=-1,rear=-1;
void main()
{
	int op;
	clrscr();
	do
	{
		printf("\n*******Main Menu*******\n");
		printf("1:Insert\n2:Delete\n3:Display\n4:Exit\n");
		printf("Enter your choice: \b");
		scanf("%d",&op);
		switch(op)
		{
			case 1:insert();break;
			case 2:delete();break;
			case 3:display();break;
			case 4:exit(0);break;
		}
	}
	while(op!=4);
	getch();
}
void insert()
{
	int data;
	if(rear==MAX-1)
	printf("\nQueue overflow\n");
```

```
        else
        {
                if(front==-1&&rear==-1)
                front=0;
                printf("\nEnter the data to insert:\n");
                scanf("%d",&data);
                queue[++rear]=data;
                printf("%d is inserted to queue succesfully\n");
        }
}
void delete()
{
        if(front==-1||front>rear)
           printf("Queue underflow\n");
        else
                printf("the deleted value:%d",queue[front++]);
}
void display()
{
        int i;
        printf("\nthe elements in queue are\n");
        for(i=front;i<=rear;i++)
         printf("%d\n",queue[i]);
}
```

Question no:23

Implement Queue using linked list

Source code :

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
struct queue
{
	int data;
	struct queue *next;
};
typedef struct queue queue;
void insert();
void delete();
void display();
void Exit();
queue *front=NULL,*rear=NULL;
void main()
{
	int op;
	clrscr();
	do
	{
		printf("\n*******Main Menu*******\n");
		printf("1:Insert\n2:Delete\n3:Display\n4:Exit\n");
		printf("Enter your choice: \b");
		scanf("%d",&op);
		switch(op)
		{
			case 1:insert();break;
			case 2:delete();break;
			case 3:display();break;
			case 4:Exit();break;
		}
	}
	while(op!=4);
	getch();
}
```

```
void insert()
{
        int data;
        queue *new_node;
        printf("\nEnter the value to insert:\n");
        scanf("%d",&data);
        new_node=(queue*)malloc(sizeof(queue));
        new_node->data=data;
        new_node->next=NULL;
        if(front==NULL&&rear==NULL)
        {
         front=new_node;
         rear=new_node;
    }
        else
        {
         rear->next=new_node;
         rear=rear->next;
    }
    printf("\n%d is pushed to stack succesfully",rear->data);
}
void delete()
{
        queue *ptr;
        if(front==NULL)
        printf("\nQueue is empty\n");
        else
        {
                ptr=front;
                printf("\n The value being deleted is : %d", ptr -> data);
                front=front->next;
                free(ptr);
        }
}
void display()
{
        queue *ptr;
        if(front==NULL)
        printf("\nQueue is empty\n");
        else
```

```
	{
		printf("\n The elements in array are\n");
		for(ptr=front;ptr!=NULL;ptr=ptr->next)
		printf("%d\n",ptr->data);
	}
}
void Exit()
{
	if(front!=NULL)
	{
		while(front!=NULL)
		delete();
	}
	exit(0);
}
```

Question no:24

Implement a binary search tree of characters.

Source Code: .

```
#include<stdio.h>
#include<stdlib.h>
#include<malloc.h>
#include<conio.h>
#include<string.h>
struct node
{
	char data;
	struct node *left;
	struct node *right;
};
typedef struct node node;
node *root=NULL;
node *insert(node*);
void display(node*);
void main()
{
	int op;
	clrscr();
	do
	{
		printf("\n******Main Menu******");
		printf("\n1:insert\n2:Display\n3:Exit");
		printf("\nchoice:\b");
		scanf("%d",&op);
		switch(op)
		{
			case 1:root=insert(root);break;
			case 2:display(root);break;
			case 3:exit(0);break;
			default:printf("\ninvalid input");
		}
```

```
        }while(op!=3);
        getch();
}
node *insert(node *root)
{
        char ch;
        int op;
        node *current,*parent,*tempnode;
        do
        {
          tempnode=(node*)malloc(sizeof(node));
          printf("\nEnter the data to insert:\n");
          scanf("%c",&ch);
          tempnode->data=ch;
          tempnode->left=NULL;
          tempnode->right=NULL;
          if(root==NULL)
          root=tempnode;
          else
          {
                current=root;
                while(current!=NULL)
                {
                        parent=current;
                        if(ch < parent->data)
                        {
                          current=current->left;
                          if(current==NULL)
                                parent->left=tempnode;
                        }
                        else
                        {
                                current=current->right;
                                if(current==NULL)
                                  parent->right=tempnode;
```

```
                    }
                }
               }

                    printf("\nDo you want to add a new
node:\n1:Yes\n2:No\nchoice::\b");
                    scanf("%d",&op);
        }while(op==1);
        return root;
}
void display(node *root)
{
        if(root!=NULL)
        {
                printf("\n%c",root->data);
                display(root->left);
                display(root->right);
        }
}
```

Question no:25

Traverse a binary search tree non recursively in preorder

Source Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <malloc.h>
#include<conio.h>
struct node
{
        int data;
        struct node *left;
        struct node *right;
};
typedef struct node node;
node *root=NULL;
node *stack[20];
int top=-1;
node *insert(node*);
void *push(node*);
node *pop();
void display(node*);
void main()
{
        int op;
        clrscr();
        do
        {
                printf("\n******Main Menu******");
                printf("\n1:insert\n2:Display\n3:Exit");
                printf("\nchoice:\b");
                scanf("%d",&op);
                switch(op)
                {
                        case 1:root=insert(root);break;
                        case 2:display(root);break;
```

```
                    case 3:exit(0);break;
                    default:printf("\ninvalid input");
                }
        }while(op!=3);
        getch();
}
node *insert(node *root)
{
        int data,op;
        node *current,*parent,*tempnode;
        do
        {
          tempnode=(node*)malloc(sizeof(node));
          printf("\nEnter the data to insert:");
          scanf("%d",&data);
          tempnode->data=data;
          tempnode->left=NULL;
          tempnode->right=NULL;
          if(root==NULL)
          root=tempnode;
          else
          {
                current=root;
                while(current!=NULL)
                {
                    parent=current;
                    if(data < parent->data)
                    {
                      current=current->left;
                      if(current==NULL)
                            parent->left=tempnode;
                    }
                    else
                    {
                            current=current->right;
```

```
                                if(current==NULL)
                                  parent->right=tempnode;
                        }
                }
               }

                        printf("\nDo you want to add a new
node:\n1:Yes\n2:No\nchoice::\b");
                        scanf("%d",&op);
        }while(op==1);
        return root;
}
void display(node *root)
{
        node *ptr;
        ptr=root;
        if(ptr==NULL)
        {
                printf("\nTree is empty\n");
                return;
        }
        push(root);
        while(top!=-1)
        {
                ptr=pop();
                if(ptr!=NULL)
                {
                  printf("%d\n",ptr->data);
                  push(ptr->right);
                  push(ptr->left);
           }
        }
}
void *push(node *ptr)
{
        stack[++top]=ptr;
}
node *pop()
{
        return stack[top--];
}
```

Question no:26

Traverse a binary search tree non recursively in inorder

Source Code: .

```
#include<stdio.h>
#include<stdlib.h>
#include<malloc.h>
#include<conio.h>
struct node
{
	int data;
	struct node *left;
	struct node *right;
};
typedef struct node node;
node *root=NULL;
node *stack[20];
int top=-1;
node *insert(node*);
void *push(node*);
node *pop();
void display(node*);
void main()
{
	int op;
	clrscr();
	do
	{
		printf("\n******Main Menu******");
		printf("\n1:insert\n2:Display\n3:Exit");
		printf("\nchoice:\b");
		scanf("%d",&op);
		switch(op)
		{
			case 1:root=insert(root);break;
			case 2:display(root);break;
```

```
                        case 3:exit(0);break;
                        default:printf("\ninvalid input");
                }
        }while(op!=3);
        getch();
}
node *insert(node *root)
{
        int data,op;
        node *current,*parent,*tempnode;
        do
        {
          tempnode=(node*)malloc(sizeof(node));
          printf("\nEnter the data to insert:");
          scanf("%d",&data);
          tempnode->data=data;
          tempnode->left=NULL;
          tempnode->right=NULL;
          if(root==NULL)
          root=tempnode;
          else
          {
                current=root;
                while(current!=NULL)
                {
                        parent=current;
                        if(data < parent->data)
                        {
                          current=current->left;
                          if(current==NULL)
                                parent->left=tempnode;
                        }
                        else
                        {
                                current=current->right;
```

```
                              if(current==NULL)
                                parent->right=tempnode;
                         }
                     }
                    }

                         printf("\nDo you want to add a new
node:\n1:Yes\n2:No\nchoice::\b");
                         scanf("%d",&op);
          }while(op==1);
          return root;
}
void display(node *root)
{
          node *ptr;
          ptr=root;
          if(ptr==NULL)
          {
                    printf("\nTree is empty\n");
                    return;
          }
          push(root);
          while(top!=-1)
          {
                    if(ptr!=NULL)
                    {
                       push(ptr->right);
                       push(ptr->left);
              }
              else
              {
                         ptr=pop();
                   printf("%d\n",ptr->data);
                    }
```

```
	}
}
void *push(node *ptr)
{
	stack[++top]=ptr;
}
node *pop()
{
	return stack[top--];
}
```

Question no:27

Traverse a binary search tree non recursively in postorder

Source Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <malloc.h>
#include<conio.h>
struct node
{
	int data;
	struct node *left;
	struct node *right;
};
typedef struct node node;
node *root=NULL;
node *stack[20];
int top=-1;
node *insert(node*);
void *push(node*);
node *pop();
node *peak();
void display(node*);
void main()
{
	int op;
	clrscr();
	do
	{
		printf("\n******Main Menu******");
		printf("\n1:insert\n2:Display\n3:Exit");
		printf("\nchoice:\b");
		scanf("%d",&op);
		switch(op)
		{
			case 1:root=insert(root);break;
```

```
                    case 2:display(root);break;
                    case 3:exit(0);break;
                    default:printf("\ninvalid input");
                }
        }while(op!=3);
        getch();
}
node *insert(node *root)
{
        int data,op;
        node *current,*parent,*tempnode;
        do
        {
          tempnode=(node*)malloc(sizeof(node));
          printf("\nEnter the data to insert:");
          scanf("%d",&data);
          tempnode->data=data;
          tempnode->left=NULL;
          tempnode->right=NULL;
          if(root==NULL)
          root=tempnode;
          else
          {
                current=root;
                while(current!=NULL)
                {
                    parent=current;
                    if(data < parent->data)
                    {
                      current=current->left;
                      if(current==NULL)
                            parent->left=tempnode;
                    }
                    else
                    {
```

```
                                        current=current->right;
                                        if(current==NULL)
                                          parent->right=tempnode;
                                }
                        }
                      }

                                printf("\nDo you want to add a new
node:\n1:Yes\n2:No\nchoice::\b");
                                scanf("%d",&op);
        }while(op==1);
        return root;
}
void display(node *root)
{
        node *ptr;
        ptr=root;
        if(ptr==NULL)
        {
                printf("\nTree is empty\n");
                return;
        }
        do
        {
                while(ptr!=NULL)
                {
                  if(ptr->right!=NULL)
                                push(ptr->right);
                  push(ptr);
                  ptr=ptr->left;
            }
            ptr=pop();
            if(ptr->right!=NULL && peak()==ptr->right)
            {
                        pop();
                        push(ptr);
                        ptr=ptr->right;
                }
                else
             {
                   printf("%d\n",ptr->data);
                   ptr=NULL;
                }
        }
        while(top!=-1);
}
void *push(node *ptr)
{
        stack[++top]=ptr;
}
node *pop()
{
        return stack[top--];
}
node *peak()
{
        return stack[top];
}
```

Question no:28

Traverse a binary search tree recursively in preorder

Source Code:
```
#include <stdio.h>
#include <stdlib.h>
#include <malloc.h>
#include<conio.h>
struct node
{
	int data;
	struct node *left;
	struct node *right;
};
typedef struct node node;
node *root=NULL;
node *insert(node*);
void display(node*);
void main()
{
	int op;
	clrscr();
	do
	{
		printf("\n******Main Menu******");
		printf("\n1:insert\n2:Display\n3:Exit");
		printf("\nchoice:\b");
		scanf("%d",&op);
		switch(op)
		{
			case 1:root=insert(root);break;
			case 2:display(root);break;
			case 3:exit(0);break;
			default:printf("\ninvalid input");
		}
	}while(op!=3);
```

```
        getch();
}
node *insert(node *root)
{
        int data,op;
        node *current,*parent,*tempnode;
        do
        {
          tempnode=(node*)malloc(sizeof(node));
          printf("\nEnter the data to insert:");
          scanf("%d",&data);
          tempnode->data=data;
          tempnode->left=NULL;
          tempnode->right=NULL;
          if(root==NULL)
          root=tempnode;
          else
          {
                current=root;
                while(current!=NULL)
                {
                        parent=current;
                        if(data < parent->data)
                        {
                          current=current->left;
                          if(current==NULL)
                                parent->left=tempnode;
                        }
                        else
                        {
                                current=current->right;
                                if(current==NULL)
                                  parent->right=tempnode;
                        }
                }
                }

                        printf("\nDo you want to add a new
node:\n1:Yes\n2:No\nchoice::\b");
                        scanf("%d",&op);
        }while(op==1);
        return root;
}
void display(node *root)
{
        if(root!=NULL)
        {
                printf("\n%d",root->data);
                display(root->left);
                display(root->right);
        }
}
```

Question no:29

Traverse a binary search tree recursively in inorder

Source Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <malloc.h>
#include<conio.h>
struct node
{
	int data;
	struct node *left;
	struct node *right;
};
typedef struct node node;
node *root=NULL;
node *insert(node*);
void display(node*);
void main()
{
	int op;
	clrscr();
	do
	{
		printf("\n******Main Menu******");
		printf("\n1:insert\n2:Display\n3:Exit");
		printf("\nchoice:\b");
		scanf("%d",&op);
		switch(op)
		{
			case 1:root=insert(root);break;
			case 2:display(root);break;
			case 3:exit(0);break;
			default:printf("\ninvalid input");
		}
	}while(op!=3);
```

```
        getch();
}
node *insert(node *root)
{
        int data,op;
        node *current,*parent,*tempnode;
        do
        {
          tempnode=(node*)malloc(sizeof(node));
          printf("\nEnter the data to insert:");
          scanf("%d",&data);
          tempnode->data=data;
          tempnode->left=NULL;
          tempnode->right=NULL;
          if(root==NULL)
          root=tempnode;
          else
          {
                current=root;
                while(current!=NULL)
                {
                        parent=current;
                        if(data < parent->data)
                        {
                          current=current->left;
                          if(current==NULL)
                                parent->left=tempnode;
                        }
                        else
                        {
                                current=current->right;
                                if(current==NULL)
                                  parent->right=tempnode;
                        }
                 }
                 }
                printf("\nDo you want to add a new node:”);
                printf(“\n1:Yes\n2:No\nchoice::\b");
                scanf("%d",&op);
        }
        while(op==1);
        return root;
}
void display(node *root)
{
        if(root!=NULL)
        {
                display(root->left);
                printf("\n%d",root->data);
                display(root->right);
        }
}
```

Question no:30

Traverse a binary search tree recursively in preorder

Source Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <malloc.h>
#include<conio.h>
struct node
{
	int data;
	struct node *left;
	struct node *right;
};
typedef struct node node;
node *root=NULL;
node *insert(node*);
void display(node*);
void main()
{
	int op;
	clrscr();
	do
	{
		printf("\n******Main Menu******");
		printf("\n1:insert\n2:Display\n3:Exit");
		printf("\nchoice:\b");
		scanf("%d",&op);
		switch(op)
		{
			case 1:root=insert(root);break;
			case 2:display(root);break;
			case 3:exit(0);break;
			default:printf("\ninvalid input");
		}
	}while(op!=3);
```

```
        getch();
}
node *insert(node *root)
{
        int data,op;
        node *current,*parent,*tempnode;
        do
        {
          tempnode=(node*)malloc(sizeof(node));
          printf("\nEnter the data to insert:");
          scanf("%d",&data);
          tempnode->data=data;
          tempnode->left=NULL;
          tempnode->right=NULL;
          if(root==NULL)
          root=tempnode;
          else
          {
                current=root;
                while(current!=NULL)
                {
                        parent=current;
                        if(data < parent->data)
                        {
                          current=current->left;
                          if(current==NULL)
                                parent->left=tempnode;
                        }
                        else
                        {
                                current=current->right;
                                if(current==NULL)
                                  parent->right=tempnode;
                        }
                }
                }

                        printf("\nDo you want to add a new 
node:\n1:Yes\n2:No\nchoice::\b");
                        scanf("%d",&op);
        }while(op==1);
        return root;
}
void display(node *root)
{
        if(root!=NULL)
        {
                display(root->left);
                display(root->right);
                printf("\n%d",root->data);
        }
}
```

Question no:31

Delete an element from a binary search tree

Source Code: .

```
#include <stdio.h>
#include <stdlib.h>
#include <malloc.h>
#include <conio.h>
struct node
{
	int data;
	struct node *left;
	struct node *right;
};
typedef struct node node;
node *root=NULL;
node *insert(node*);
node *successor(node*);
node *delete(node*,int );
void display(node*);
void check(void (*fn)(node *),node *root)
{
	if(root==NULL)
	printf("\nThe list is empty");
	else
	fn(root);
}
void main()
{
	int op;
	clrscr();
	do
	{
		printf("\n******Main Menu******");
		printf("\n1:insert\n2:Display\n3:Delete\n4:Exit");
		printf("\nchoice:\b");
```

```
                scanf("%d",&op);
                switch(op)
                {
                        case 1:root=insert(root);break;
                        case 2:check(display,root);break;
                        case 3:root=delete(root,-1);break;
                        case 4:exit(0);break;
                        default:printf("\ninvalid input");
                }
        }while(op!=4);
        getch();
}
node *insert(node *root)
{
        int data,op;
        node *current,*parent,*tempnode;
        do
        {
          tempnode=(node*)malloc(sizeof(node));
          printf("\nEnter the data to insert:");
          scanf("%d",&data);
          tempnode->data=data;
          tempnode->left=NULL;
          tempnode->right=NULL;
          if(root==NULL)
          root=tempnode;
          else
          {
                current=root;
                while(current!=NULL)
                {
                        parent=current;
                        if(data < parent->data)
                        {
                          current=current->left;
                          if(current==NULL)
                                parent->left=tempnode;
                        }
                        else
                        {
                                current=current->right;
                                if(current==NULL)
                                  parent->right=tempnode;
                        }
                }
                }

                        printf("\nDo you want to add a new
node:\n1:Yes\n2:No\nchoice::\b");
                        scanf("%d",&op);
        }while(op==1);
        return root;
}
void display(node *root)
{
        if(root!=NULL)
        {
                printf("\n%d",root->data);
                display(root->left);
                display(root->right);
        }
}
node *delete(node *root,int n)
{
        int data,i;
        node *ptr,*pt=NULL,*ptr1=NULL;
        ptr=root;
        int flag=0;
        if(n==-1)
        {
```

```
        printf("\nEnter the value to delete:");
        scanf("%d",&data);
}
else
        data=n;
while(ptr!=NULL&&flag==0)
{
        if(data<ptr->data)
        {
                pt=ptr;
                ptr=ptr->left;
        }
        else if(data==ptr->data)
                flag=1;
        else
        {
                pt=ptr;
                ptr=ptr->right;
        }
}
if(flag==0)
  printf("\nThe element is not found in the tree:\n");
else
{
        if(ptr->left==NULL&&ptr->right==NULL)
        {
                if(pt->left==ptr)
                        pt->left=NULL;
                if(pt->right==ptr)
                        pt->right=NULL;
                free(ptr);
        }
        else if(ptr->left!=NULL&&ptr->right!=NULL)
        {
                ptr1=successor(ptr);
```

```
                    i=ptr1->data;
                    root=delete(root,i);
                    ptr->data=i;
                }
                else
                {
                    if(pt->left==ptr)
                    {
                        if(ptr->left==NULL)
                            pt->left=ptr->right;
                        else
                            pt->left=ptr->left;
                    }
                    if(pt->right==ptr)
                    {
                        if(ptr->left==NULL)
                            pt->right=ptr->right;
                        else
                            pt->right=ptr->left;
                    }
                    free(ptr);          }
            }
            return root;
}
node *successor(node *ptr)
{
        node *p;
        p=ptr->right;
        if(p!=NULL)
                {
                    while(p->left!=NULL)
                        p=p->left;
                }
        return p;
}
```

Question no:32

Search an element from a binary search tree

Source Code: .

```
#include <stdio.h>
#include <stdlib.h>
#include <malloc.h>
#include <conio.h>
struct node
{
	int data;
	struct node *left;
	struct node *right;
};
typedef struct node node;
node *root=NULL;
node *insert(node*);
void display(node*);
void check(void (*fn)(node *),node *root)
{
	if(root==NULL)
	printf("\nThe list is empty");
	else
	fn(root);
}
void search(node*);
void main()
{
	int op;
	clrscr();
	do
	{
		printf("\n******Main Menu******");
		printf("\n1:insert\n2:Display\n3:Search\n4:Exit");
		printf("\nchoice:\b");
		scanf("%d",&op);
```

```
                switch(op)
                {
                        case 1:root=insert(root);break;
                        case 2:check(display,root);break;
                        case 3:check(search,root);break;
                        case 4:exit(0);break;
                        default:printf("\ninvalid input");
                }
        }while(op!=3);
        getch();
}
node *insert(node *root)
{
        int data,op;
        node *current,*parent,*tempnode;
        do
        {
          tempnode=(node*)malloc(sizeof(node));
          printf("\nEnter the data to insert:");
          scanf("%d",&data);
          tempnode->data=data;
          tempnode->left=NULL;
          tempnode->right=NULL;
          if(root==NULL)
          root=tempnode;
          else
          {
                current=root;
                while(current!=NULL)
                {
                        parent=current;
                        if(data < parent->data)
                        {
                          current=current->left;
                          if(current==NULL)
                            parent->left=tempnode;
                        }
                        else
                        {
                            current=current->right;
                            if(current==NULL)
                              parent->right=tempnode;
                        }
                }
                }
                printf("\nDo you want to add a new node:”);
                printf(“\n1:Yes\n2:No\nchoice::\b");
                scanf("%d",&op);
        }
        while(op==1);
        return root;
}
void display(node *root)
{
        if(root!=NULL)
        {
                printf("\n%d",root->data);
                display(root->left);
                display(root->right);
        }
}
void search(node *root)
{
        node *ptr;
        int item,flag=0;
        ptr=root;
        printf("\nEnter the value to search:");
        scanf("%d",&item);
        while(ptr!=NULL&&flag==0)
        {
                if(item<ptr->data)
                            ptr=ptr->left;
                else if(item==ptr->data)
                            flag=1;
                else
                            ptr=ptr->right;
        }
        if(flag==1)
          printf("\nThe element is found on the tree \n");
        else
                printf("\nThe item is doesn't exist in the tree \n");
}
```

Question no:34

Implement bubble sort

Source Code: .

```
#include<stdio.h>
#include<conio.h>
void main()
{
  int a[50],n,i,temp,j;
  clrscr();
  printf("\nEntere the size of array:");
  scanf("%d",&n);
  printf("\nEnter the Elements to array:\n");
  for(i=0;i<n;i++)
        scanf("%d",&a[i]);
  for(i=0;i<n;i++)
  {
    for(j=0;j<n-i;j++)
    {
      if(a[j]>a[j+1])
      {
        temp=a[j];
        a[j]=a[j+1];
        a[j+1]=temp;
      }
    }
  }
  printf("\nThe sorted array is:\n");
  for(i=0;i<n;i++)
        printf("%d\n",a[i]);
  getch();
}
```

Question no:35

Implement exchange sort

Source code :

```
#include <stdio.h>
#include <conio.h>
void main()
{
	int a[100],i,j,temp,n;
	clrscr();
	printf("\nEnter the size of array:");
	scanf("%d",&n);
	printf("\nEnter the array:\n");
	for(i=0;i<n;i++)
		scanf("%d",&a[i]);
	for(i=0;i<n-1;i++)
	{
		for(j=i+1;j<n;j++)
		{
			if(a[i]>a[j])
			{
				temp=a[i];
				a[i]=a[j];
				a[j]=temp;
			}
		}
	}
	printf("\nThe sorted array is :\n");
	for(i=0;i<n;i++)
		printf("%d\n",a[i]);
	getch();
}
```

Question no:36

Implement selection sort

Source code :

```
#include <stdio.h>
#include <stdlib.h>
int smallest(int arr[], int k, int n);
void selection_sort(int arr[], int n);
int main()
{
	int arr[100],i, n;
	printf("\n Enter the number of elements in the array: ");
	scanf("%d", &n);
	printf("\n Enter the elements: ");
	for(i=0;i<n;i++)
		scanf("%d", &arr[i]);
	selection_sort(arr, n);
	printf("\n The sorted array is: \n");
	for(i=0;i<n;i++)
		printf("%d\n",arr[i]);
	return 0;
}
int smallest(int arr[], int k, int n)
{
	int pos = k, small=arr[k], i;
	for(i=k+1;i<n;i++)
	{
		if(arr[i]< small)
		{
			small = arr[i];
			pos = i;
		}
	}
	return pos;
}
```

```
void selection_sort(int arr[],int n)
{
        int k, pos, temp;
        for(k=0;k<n;k++)
        {
                pos = smallest(arr, k, n);
                temp = arr[k];
                arr[k] = arr[pos];
                arr[pos] = temp;
        }
}
```

Question no:37
Implement insertion sort

Source code :
```
#include <stdio.h>
#include <conio.h>
void insertionSort(int arr[],int n);
void main()
{
   int arr[100],n,i;
   clrscr();
   printf("\nEnter the sizeof the array:\n");
   scanf("%d",&n);
   printf("\nEnter the Elements to the array:\n");
   for(i=0;i<n;i++)
               scanf("%d",&arr[i]);
   insertionSort(arr,n);
   printf("Sorted array: \n");
   for(i=0;i<n;i++)
               printf("%d\n",arr[i]);
   getch();
}
void insertionSort(int arr[],int n)
{
        int i, j, temp;
        for(i=1;i<n;i++)
        {
               temp = arr[i];
               j = i-1;
               while((temp < arr[j]) && (j>=0))
               {
                      arr[j+1] = arr[j];
                      j--;
               }
               arr[j+1] = temp;
        }
    }
```

Question no:38

Implement quick sort

Source code :

```
#include <stdio.h>
#include <conio.h>
void quickSort(int arr[],int first,int last)
{
	int p;
	if(first<last)
	{
		p=partition(arr,first,last);
		quickSort(arr,first,p-1);
		quickSort(arr,p+1,last);
	}
}
int partition(int a[],int beg,int end)
{
	int left, right, temp, loc, flag;
	loc = left = beg;
	right = end;
	flag = 0;
	while(flag != 1)
	{
		while((a[loc] <= a[right]) && (loc!=right))
		right--;
		if(loc==right)
		flag =1;
		else if(a[loc]>a[right])
		{
			temp = a[loc];
			a[loc] = a[right];
			a[right] = temp;
			loc = right;
		}
```

```
                if(flag!=1)
                {
                        while((a[loc] >= a[left]) && (loc!=left))
                                left++;
                        if(loc==left)
                                flag=1;
                        else if(a[loc] <a[left])
                        {
                                temp = a[loc];
                                a[loc] = a[left];
                                a[left] = temp;
                                loc = left;
                        }
                }
        }
        return loc;
}
void main()
{
   int arr[100],n,i;
   clrscr();
   printf("\nEnter the sizeof the array:\n");
   scanf("%d",&n);
   printf("\nEnter the Elements to the array:\n");
   for(i=0;i<n;i++)
                scanf("%d",&arr[i]);
   quickSort(arr, 0, n-1);
   printf("Sorted array: \n");
   for(i=0;i<n;i++)
                printf("%d\n",arr[i]);
   getch();
}
```

Question no:39

Implement merge sort

Source code :

```
#include <stdio.h>
#include <conio.h>
void mergeSort(int arr[],int first,int last);
void merge(int arr[],int beg,int mid,int end);
void main()
{
   int arr[100],n,i;
   clrscr();
   printf("\nEnter the sizeof the array:\n");
   scanf("%d",&n);
   printf("\nEnter the Elements to the array:\n");
   for(i=0;i<n;i++)
               scanf("%d",&arr[i]);
   mergeSort(arr, 0, n-1);
   printf("Sorted array: \n");
   for(i=0;i<n;i++)
               printf("%d\n",arr[i]);
   getch();
}
void mergeSort(int arr[],int first,int last)
{
        int p;
        if(first<last)
        {
                p=(first+last)/2;
                mergeSort(arr,first,p);
                mergeSort(arr,p+1,last);
                merge(arr,first,p,last);
        }
}
```

```
void merge(int arr[],int beg,int mid,int end)
{
        int i=beg,j=mid+1,index=beg,temp[100],k;
        while((i<=mid) && (j<=end))
        {
                if(arr[i] < arr[j])
                {
                        temp[index] = arr[i];
                        i++;
                }
                else
                {
                        temp[index] = arr[j];
                        j++;
                }
                index++;
        }
        if(i>mid)
        {
                while(j<=end)
                {
                        temp[index] = arr[j];
                        j++;
                        index++;
                }
        }
        else
        {
                while(i<=mid)
                {
                        temp[index] = arr[i];
                        i++;
                        index++;
                }
        }
        for(k=beg;k<index;k++)
                arr[k]=temp[k];
}
```

Question no:40

Implement heap sort

Source code :

```
#include <stdio.h>
#include <conio.h>
void RestoreHeapUp(int *,int);
void RestoreHeapDown(int*,int,int);
void main()
{
	int Heap[100],n,i,j;
	clrscr();
	printf("\n Enter the number of elements : ");
	scanf("%d",&n);
	printf("\n Enter the elements : ");
	for(i=1;i<=n;i++)
	{
		scanf("%d",&Heap[i]);
		RestoreHeapUp(Heap, i);
	}
	j=n;
	for(i=1;i<=j;i++)
	{
		int temp;
		temp=Heap[1];
		Heap[1]= Heap[n];
		Heap[n]=temp;
		n--;
		RestoreHeapDown(Heap,1,n);
	}
	n=j;
	printf("\n The sorted elements are: ");
	for(i=1;i<=n;i++)
		printf("%4d",Heap[i]);
	getch();
}
void RestoreHeapUp(int *Heap,int index)
{
        int val = Heap[index];
        while( (index>1) && (Heap[index/2] < val) )
        {
                Heap[index]=Heap[index/2];
                index /= 2;
        }
        Heap[index]=val;
}
void RestoreHeapDown(int *Heap,int index,int n)
{
        int val = Heap[index];
        int j=index*2;
        while(j<=n)
        {
                if( (j<n) && (Heap[j] < Heap[j+1]))
                        j++;
                if(Heap[j] < Heap[j/2])
                        break;
                Heap[j/2]=Heap[j];
                j=j*2;
        }
        Heap[j/2]=val;
}
```