Program#1
Write programs to demonstrate the usage of storage classes in C

Source Code:

```
#include<stdio.h>
int gv=5;
void staticfn()
{
        static int sv=100;
        printf("Value of static variable is :%d\n",sv);
        sv++;
}
int main()
{
        int lv=10;
        register int rv=50;
        printf("Inside main function\n");
        printf("Value of global variable is :%d\n",gv);
        printf("Value of local variable is :%d\n",lv);
        printf("Value of register variable is :%d\n",rv);
        staticfn();

        {
                int lv=20;
                gv++;
                printf("Inside block A\n");
                printf("Value of global variable is :%d\n",gv);
                printf("Value of local variable is :%d\n",lv);
                staticfn();
        }
        {
                gv++;
                printf("Inside Block B\n");
                printf("Value of global variable is :%d\n",gv);
                printf("Value of local variable is :%d\n",lv);
                staticfn();
        }
        return 0;
}
```

Output

```
Inside main function
Value of global variable is :5
Value of local variable is :10
Value of register variable is :50
Value of static variable is :100
Inside block A
Value of global variable is :6
Value of local variable is :20
Value of static variable is :101
Inside Block B
Value of global variable is :7
Value of local variable is :10
Value of static variable is :102

--------------------------------
Process exited after 0.4938 seconds with return value 0
Press any key to continue . . .
```

Program#2
Allocate a two-dimensional array dynamically.

Source Code:

```
#include<stdio.h>
#include<malloc.h>
int main()
{
        int i,j;
        int r=2;
        int c=3;
        int *p=(int *)malloc(r*c*sizeof(int));
        printf("Enter the elements : ");
        for(i=0;i<r;i++)
        {
                for(j=0;j<c;j++)
                {
                        scanf("%d",p + (i*c + j));
                }
        }

        printf("\n");
        printf("\n");
        for(i=0;i<r;i++)
        {
                for(j=0;j<c;j++)
                {
                        printf("%d\t",*(p + (i*c + j)));
                }
                printf("\n");
        }

        return 0;

}
```

Output

```
Enter the elements :
1
2
3
4
5
6


1       2       3
4       5       6
```

Program#3
Use a menu driven program to insert, search, delete and sort elements in an array using functions (use global variables)

Source Code:

```
#include<stdio.h>
#define SIZE 5
int a[SIZE],pos=-1;
void insert()
{
	if(pos==SIZE-1)
	{
		printf("Array is full\n");

	}
	else
	{
		printf("Enter element to be inserted\n");
		pos++;
		scanf("%d",&a[pos]);

	}
}
void delete()
{
	if(pos==-1)
	{
		printf("Array is empty\n");
	}
	else
	{
		int p,i;
		printf("\nEnter position to be deleted\n");
		scanf("%d",&p);
		if((p>pos)||(p<0))
		{
			printf("Wrng position\n");
		}
		else
		{
			for(i=p;i<pos;i++)
			{
				a[i]=a[i+1];
			}
			pos--;
			printf("Deleted successfully\n");
		}
```

```
        }
}

void display()
{
        int i;
        printf("\nThe elements are\n");
        for(i=0;i<=pos;i++)
        {
                printf("%d\t",a[i]);
        }
        printf("\n");
}
void sort()
{
int i,j,temp;
for(i=0;i<pos;i++)
{
        for(j=i+1;j<=pos;j++)
        {
                if(a[i]>a[j])
                {
                        temp=a[i];
                        a[i]=a[j];
                        a[j]=temp;
                }
        }
}
display();
}
void search()
{
        int flag=0,num,i;
        printf("Enter element to be search\n");
        scanf("%d",&num);
        for(i=0;i<=pos;i++)
        {
                if(num==a[i])
                {
                        flag=1;
                        printf("Elemet found at position : %d\n",i);
                        break;
                }
        }
        if(flag==0)
        {
```

```
                printf("Element not found\n");
        }
}
int menu()
{
        int ch;
        printf("\n1.Insert\n2.Delete\n3.Sort\n4.Display\n5.Search\n6.Exit\nEnter your choice
: ");
        scanf("%d",&ch);
        return ch;
}
int main()
{
        int ch;
        ch = menu();
        while(ch!=6)
        {
                switch(ch)
                {
                        case 1:
                                insert();
                                break;
                        case 2:
                                delete();
                                break;
                        case 3:
                                sort();
                                break;
                        case 4:
                                display();
                                break;
                        case 5:
                                search();
                                break;
                        default:
                                printf("Wrong choice\n");
                                break;
                }
                ch=menu();
        }

        return 0;
}
```

Output

```
1.Insert
2.Delete
3.Sort
4.Display
5.Search
6.Exit
Enter your choice : 1
Enter element to be inserted
5

1.Insert
2.Delete
3.Sort
4.Display
5.Search
6.Exit
Enter your choice : 1
Enter element to be inserted
4

1.Insert
2.Delete
3.Sort
4.Display
5.Search
6.Exit
Enter your choice : 1
Enter element to be inserted
10

1.Insert
2.Delete
3.Sort
4.Display
5.Search
6.Exit
Enter your choice : 1
Enter element to be inserted
9
```

```
1.Insert
2.Delete
3.Sort
4.Display
5.Search
6.Exit
Enter your choice : 1
Enter element to be inserted
90

1.Insert
2.Delete
3.Sort
4.Display
5.Search
6.Exit
Enter your choice : 4

The elements are
5       4       10      9       90

1.Insert
2.Delete
3.Sort
4.Display
5.Search
6.Exit
Enter your choice : 2

Enter position to be deleted
4
Deleted successfully

1.Insert
2.Delete
3.Sort
4.Display
5.Search
6.Exit
Enter your choice : 3
```

```
Enter your choice : 3

The elements are
4       5       9       10

1.Insert
2.Delete
3.Sort
4.Display
5.Search
6.Exit
Enter your choice : 5
Enter element to be search
5
Elemet found at position : 1

1.Insert
2.Delete
3.Sort
4.Display
5.Search
6.Exit
```

Program#4
Use a menu driven program to insert, search, delete and sort elements in an array using functions (use only local variables)

Source Code:

```
#include<stdio.h>
#define SIZE 10
int insert(int arr[],int pos)
{
	if(pos==SIZE-1)
	{
		printf("Array is full\n");

	}
	else
	{
		printf("Enter element to be inserted\n");
		pos++;
		scanf("%d",&arr[pos]);

	}
	return pos;
}
int delete(int a[],int pos)
{
	if(pos==-1)
	{
		printf("Array is empty\n");
	}
	else
	{
		int p,i;
		printf("\nEnter position to be deleted\n");
		scanf("%d",&p);
		if((p>pos)||(p<0))
		{
			printf("Wrng position\n");
		}
		else
		{
			for(i=p;i<pos;i++)
			{
				a[i]=a[i+1];
			}
			pos--;  //pos must be decrement by 1(because when deleted the pos is
decreades by 1)
			printf("Deleted successfully\n");
```

```
		}
	}
	return pos;
}

void display(int a[],int pos)
{
	int i;
	printf("\nThe elements are\n");
	for(i=0;i<=pos;i++)
	{
		printf("%d\t",a[i]);
	}
	printf("\n");
}
void sort(int a[],int pos)
{
int i,j,temp;
for(i=0;i<pos;i++)
{
	for(j=i+1;j<=pos;j++)
	{
		if(a[i]>a[j])
		{
			temp=a[i];
			a[i]=a[j];
			a[j]=temp;
		}
	}
}
display(a,pos);
}
void search(int a[],int pos)
{
	int flag=0,num,i;
	printf("Enter element to be search\n");
	scanf("%d",&num);
	for(i=0;i<=pos;i++)
	{
		if(num==a[i])
		{
			flag=1;
			printf("Element found at position : %d\n",i);
			break;
		}
	}
```

```
	if(flag==0)
	{
		printf("Element not found\n");
	}
}
int menu()
{
	int ch;
	printf("\n1.Insert\n2.Delete\n3.Sort\n4.Display\n5.Search\n6.Exit\nEnter your choice
: ");
	scanf("%d",&ch);
	return ch;
}
int main()
{
	int arr[SIZE],pos=-1;
	int ch;
	ch = menu(); //initialization
	while(ch!=6) //condition
	{
		switch(ch)
		{
			case 1:
				pos = insert(arr,pos);
				break;
			case 2:
				pos=delete(arr,pos);
				break;
			case 3:
				sort(arr,pos);
				break;
			case 4:
				display(arr,pos);
				break;
			case 5:
				search(arr,pos);
				break;
			default:
				printf("Wrong choice\n");
				break;
		}
		ch=menu(); //updation
	}

	return 0;
}
```

Output

```
1.Insert
2.Delete
3.Sort
4.Display
5.Search
6.Exit
Enter your choice : 1
Enter element to be inserted
5

1.Insert
2.Delete
3.Sort
4.Display
5.Search
6.Exit
Enter your choice : 1
Enter element to be inserted
4

1.Insert
2.Delete
3.Sort
4.Display
5.Search
6.Exit
Enter your choice : 1
Enter element to be inserted
10

1.Insert
2.Delete
3.Sort
4.Display
5.Search
6.Exit
Enter your choice : 1
Enter element to be inserted
9
```

```
1.Insert
2.Delete
3.Sort
4.Display
5.Search
6.Exit
Enter your choice : 1
Enter element to be inserted
90

1.Insert
2.Delete
3.Sort
4.Display
5.Search
6.Exit
Enter your choice : 4

The elements are
5       4       10      9       90

1.Insert
2.Delete
3.Sort
4.Display
5.Search
6.Exit
Enter your choice : 2

Enter position to be deleted
4
Deleted successfully

1.Insert
2.Delete
3.Sort
4.Display
5.Search
6.Exit
Enter your choice : 3
```

```
Enter your choice : 3

The elements are
4       5       9       10

1.Insert
2.Delete
3.Sort
4.Display
5.Search
6.Exit
Enter your choice : 5
Enter element to be search
5
Elemet found at position : 1

1.Insert
2.Delete
3.Sort
4.Display
5.Search
6.Exit
```

Program#5
Search for all the occurrence of an element in an integer array (positions)

Source Code:

```
#include<stdio.h>
int a[20],n;
void insert()
{
	int i;
	printf("Enter size of array\n");
	scanf("%d",&n);
	printf("Enter elemnts\n");
	for(i=0;i<n;i++)
	{
		scanf("%d",&a[i]);
	}

}
void occurence()
{
	int num,flag=0,i;
	printf("Enter the elemet to be searched\n");
	scanf("%d",&num);
	for(i=0;i<n;i++)
	{
		if(a[i]==num)
		{
			flag=1;
			printf("Element found at position :%d\n",i);
		}
	}
	if(flag==0)
	{
		printf("Element not found");
	}
}
int main()
{
	insert();
	occurence();
	return 0;
}
```

Output

```
Enter size of array
5
Enter elemnts
2
2
3
6
6
Enter the elemet to be searched
6
Element found at position :3
Element found at position :4
```

Program#6
Sort the array elements in ascending order (minimum three functions - read, disp and sort)

Source Code:

```
#include<stdio.h>
#define SIZE 10
int a[SIZE],i,n,j;
void read()
{
	printf("Enter size of array\n");
	scanf("%d",&n);
	printf("Enter elements\n");
	for(i=0;i<n;i++)
	{
		scanf("%d",&a[i]);

	}
}
void sort()
{
	int temp;
	for(i=0;i<n;i++)
	{
		for(j=i+1;j<=n-1;j++)
			{
				if(a[i]>a[j])
				{
					temp=a[i];
					a[i]=a[j];
					a[j]=temp;
				}

			}
	}
}
void display()
{
	sort();
	printf("Sorted elements are :");
	for(i=0;i<n;i++)
	{
		printf("%d\t",a[i]);
	}
}
int main()
{
	read();
	sort();
```

```
        display();
        return 0;
}
```

Output

```
Enter size of array
5
Enter elements
1
7
10
3
2
Sorted elements are :1  2       3       7       10
--------------------------------
```

Program#7
Two Dimensional Matrix - using functions **a. Addition** **b. Subtraction** **c. Multiplication** **d. Transpose** **e. Determinant**

Source Code:

```
#include <stdio.h>
#define N 3
void matrixAddition(int A[N][N], int B[N][N], int result[N][N]) {
        int i,j;
   for ( i = 0; i < N; i++) {
      for ( j = 0; j < N; j++) {
         result[i][j] = A[i][j] + B[i][j];
      }
   }
}

void matrixSubtraction(int A[N][N], int B[N][N], int result[N][N]) {
                int i,j;
   for ( i = 0; i < N; i++) {
      for ( j = 0; j < N; j++) {
         result[i][j] = A[i][j] - B[i][j];
      }
   }
}

void matrixMultiplication(int A[N][N], int B[N][N], int result[N][N]) {
                int i,j,k;
   for ( i = 0; i < N; i++) {
      for ( j = 0; j < N; j++) {
         result[i][j] = 0;
         for ( k = 0; k < N; k++) {
            result[i][j] += A[i][k] * B[k][j];
         }
      }
   }
}
void matrixTranspose(int A[N][N], int result[N][N]) {
int i,j;
   for ( i = 0; i < N; i++) {
      for ( j = 0; j < N; j++) {
         result[i][j] = A[j][i];
      }
   }
```

```
}

int matrixDeterminant(int A[N][N]) {
                    int i,j,col;
   int det = 0;
   if (N == 1) {
      return A[0][0];
   }
   if (N == 2) {
      return A[0][0] * A[1][1] - A[0][1] * A[1][0];
   }

   for ( col = 0; col < N; col++) {
      int submatrix[N - 1][N - 1];
      for ( i = 1; i < N; i++) {
         int subcol = 0;
         for ( j = 0; j < N; j++) {
            if (j == col) {
               continue;
            }
            submatrix[i - 1][subcol] = A[i][j];
            subcol++;
         }
      }
      int sign = (col % 2 == 0) ? 1 : -1;
      det += sign * A[0][col] * matrixDeterminant(submatrix);
   }
   return det;
}
int main() {
int i,j;
   int A[N][N] = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
   int B[N][N] = {{9, 8, 7}, {6, 5, 4}, {3, 2, 1}};
   int result[N][N];
   matrixAddition(A, B, result);
   printf("Addition Result:\n");
   for ( i = 0; i < N; i++) {
      for ( j = 0; j < N; j++) {
         printf("%d ", result[i][j]);
      }
      printf("\n");
   }
   matrixSubtraction(A, B, result);
   printf("Subtraction Result:\n");
   for ( i = 0; i < N; i++) {
      for ( j = 0; j < N; j++) {
```

```
        printf("%d ", result[i][j]);
    }
    printf("\n");
}
matrixMultiplication(A, B, result);
printf("Multiplication Result:\n");
for ( i = 0; i < N; i++) {
    for ( j = 0; j < N; j++) {
        printf("%d ", result[i][j]);
    }
    printf("\n");
}
matrixTranspose(A, result);
printf("Transpose Result:\n");
for ( i = 0; i < N; i++) {
    for ( j = 0; j < N; j++) {
        printf("%d ", result[i][j]);
    }
    printf("\n");
}
int det = matrixDeterminant(A);
printf("Determinant of Matrix A: %d\n", det);
return 0;
}
```

Output

```
Addition Result:
10 10 10
10 10 10
10 10 10
Subtraction Result:
-8 -6 -4
-2 0 2
4 6 8
Multiplication Result:
30 24 18
84 69 54
138 114 90
Transpose Result:
1 4 7
2 5 8
3 6 9

--------------------------------
```

Program#8
Allocate a two dimensional array using pointer.

Source Code:

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
int r, c, i, j, count;
printf("Enter the no of rows and columns:");
scanf("%d%d",&r,&c);
int *arr[r];
for (i=0; i<r; i++)
{
arr[i] = (int *)malloc(c * sizeof(int));
}
count = 0;
for (i = 0; i < r; i++)
{
for (j = 0; j < c; j++)
{
arr[i][j] = ++count;
}
}
for (i = 0; i < r; i++)
{
for (j = 0; j < c; j++)
{
printf("%d\t", arr[i][j]);
}
printf("\n");
}
return 0;
}
```

Output

```
Enter the no of rows and columns:3
2
1       2
3       4
5       6
```

Program#9
Display the array elements in the same order using a recursive function

Source Code:

```
#include<stdio.h>
int n;
void disp(int a[])
{
        static int i = 0;
        if(i<n)
        {
                printf("%d\t",a[i]);
                i++;
                disp(a);  //recursive statement
        }


}
int main()
{
        int a[20],i;
        printf("Enter size of elements\n");
        scanf("%d",&n);
        printf("Enter elements\n");
        for(i=0;i<n;i++)
        {
                scanf("%d",&a[i]);
        }
        disp(a);
        return 0;
}
```

Output

```
Enter size of elements
5
Enter elements
8
4
1
0
6
8       4       1       0       6
--------------------------------
```

Program#10
Display array elements in the reverse order using a recursive function

Source Code:

```
#include<stdio.h>
int n;
void disp(int a[],int i)
{
        if(i>=0)
        {
                printf("%d\t",a[i]);
                i--;
                disp(a,i);  //recursive statement
        }


}
int main()
{
        int a[20],i;
        printf("Enter size of elements\n");
        scanf("%d",&n);
        printf("Enter elements\n");
        for(i=0;i<n;i++)
        {
                scanf("%d",&a[i]);
        }
        disp(a,n-1);
        return 0;
}
```

Output

```
Enter size of elements
5
Enter elements
2
6
9
7
4
4       7       9       6       2
--------------------------------
```

Program#11
Read a String and display it in the reverse order using **a. Just print it in the reverse order** **b. Reverse the string in the same array itself**

a.Just print it in the reverse order

Source Code:

```
#include<stdio.h>
#include<string.h>
int main()
{
char s[20];
int len,i;
printf("Enter string:");
scanf("%s",s);
printf("\nThe string is : %s",s);
len=strlen(s);
printf("\nReversed string is:");
for(i=len-1;i>=0;i--)
        {
                printf("%c",s[i]);
        }
return 0;

}
```

Output

```
Enter string:english

The string is : english
Reversed string is:hsilgne
--------------------------------
```

b.Reverse the string in the same array itself

Source Code:

```
#include<stdio.h>
#include<string.h>
int main()
{
	char s[20],temp;
	int len,i,j;
	printf("Enter string ");
	scanf("%s",s);
	printf("\nThe string is : %s",s);
	len=strlen(s);
	for(i=0,j=len-1;i<=j;i++,j--)
		{
			temp=s[i];
			s[i]=s[j];
			s[j]=temp;
		}
	printf("\nThe reversed array is : %s",s);
return 0;
}
```

Output

```
Enter string statistics

The string is : statistics
The reversed array is : scitsitats
--------------------------------
```

Program#12
Read n Strings and display them in the ascending order.

Source Code:

```
#include<stdio.h>
#include<string.h>
int main()
{
char string[100][100],temp[100];
int i,n,j;
printf("Enter the Number of names:");
scanf("%d",&n);
for(i=0;i<n;i++)
{
printf("\nEnter the name %d:",i+1);
scanf("%s",string[i]);
}
printf("\nBefore sorting:\n");
for(i=0;i<n;i++)
{
printf("%s\n",string[i]);
}
for(i=0;i<n;i++)
{
for(j=i+1;j<n;j++)
{
if(strcmp(string[i],string[j])>0)
{
strcpy(temp,string[i]);
strcpy(string[i],string[j]);
strcpy(string[j],temp);
}
}
}
printf("\nAfter sorting:\n");
```

```
for(i=0;i<n;i++)
{
printf("%s\n",string[i]);
}
return 0;
}
```

Output

```
Enter the Number of names:3

Enter the name 1:Saniya

Enter the name 2:Taniya

Enter the name 3:Aniya

Before sorting:
Saniya
Taniya
Aniya

After sorting:
Aniya
Saniya
Taniya
```

Program#13
Define a structure for date having dd/mm/yyyy. Provide functions for reading, displaying and comparing two dates are equal or not

Source Code:

```
#include<stdio.h>
struct Date
{
   int day, month, year;
};
struct Date date1,date2;
void readDates() {
   printf("Enter date1 (DD/MM/YYYY) : ");
   scanf("%d/%d/%d", &date1.day, &date1.month, &date1.year);
   printf("Enter date2 (DD/MM/YYYY) : ");
   scanf("%d/%d/%d", &date2.day, &date2.month, &date2.year);
}
void printDates()
 {
   printf("Date 1 : ");
   printf("%2d/%2d/%2d\n", date1.day, date1.month, date1.year);
   printf("Date 2 : ");
   printf("%2d/%2d/%2d\n", date2.day, date2.month, date2.year);
}
int isDatesEqual(struct Date date1, struct Date date2)
{
 return (date1.day == date2.day && date1.month == date2.month && date2.year ==
date2.year);
}
int main() {
   readDates();
   printDates();
   if(isDatesEqual(date1,date2))
    printf("\nDates are same");
```

```
    else
        printf("\nDates are not same")
}
```

Output

```
Enter date1 (DD/MM/YYYY) : 01/12/2023
Enter date2 (DD/MM/YYYY) : 08/12/2023
Date 1 :  1/12/2023
Date 2 :  8/12/2023

Dates are not same
```

Program#14

Define a structure for employees having eno,ename, esal and dno. Read n employees information and provide function for the following

a. **Searching an employee by no**
b. **Sorting the employee by**
   i. **Name**
   ii. **Salary**
c. **Deleting an employee**

Source Code:

```
#include<stdio.h>
#include<string.h>
struct emp
{
   char empno[10];
   char empname[20];
   int empsalary;
   char empdno[10];
};

void read(struct emp E[], int n)
{
   int i;
   for(i=0;i<n;i++)
   {
      printf("Enter the details of Employee %d\n",i+1);
      printf("Enter the Employee No : ");
      scanf("%s",E[i].empno);
      printf("Enter the name : ");
      scanf("%s",E[i].empname);
      printf("Enter the salary : ");
      scanf("%d",&E[i].empsalary);
      printf("Enter the department no : ");
      scanf("%s",E[i].empdno);
```

```
        printf("\n");
    }
}

void disp(struct emp *E, int n)
{
    int i;

    printf("Emp no\tName\t  Salary\tDepNo\n");
    for(i=0;i<n;i++)
    {
        printf("%s\t%s\t%d\t\t%s\n",E[i].empno,E[i].empname,E[i].empsalary,E[i].empdno);
    }
    printf("\n");
}
void search(struct emp *E, int n)
{
        char eno[20];
        int i,flag=0;
        if(n==0)
                printf("List Empty!!");
        else
        {
         printf("Enter employee No to be searched : ");
         scanf("%s",eno);
         for(i=0;i<n;i++)
         {
                if(strcmp(E[i].empno,eno) ==0)
                {
                        flag=1;
                        printf("The employee is found\n");
                        printf("Emp No : %s\nName : %s\nSalary : %d\nDep No :
%s\n",E[i].empno,E[i].empname,E[i].empsalary,E[i].empdno);
```

```
                    break;
                }
            }
            if(flag == 0)
                printf("%s is not found\n",eno);
    }
}
void sortSalary(struct emp *E, int n)
{
    struct emp temp;
    int i,j;
    for(i=0;i<n-1;i++)
    {
        for(j=i+1;j<n;j++)
        {
            if(E[i].empsalary > E[j].empsalary)
            {
                temp = E[i];
                E[i] = E[j];
                E[j] = temp;
            }
        }
    }
    printf("\n\nThe employee list is successfully sorted by salary\n");
}
void sortName(struct emp *E, int n)
{
    struct emp temp;
    int i,j;
    for(i=0;i<n-1;i++)
    {
        for(j=i+1;j<n;j++)
        {
```

```
            if(strcmp(E[i].empname,E[j].empname) > 0)
            {
                temp = E[i];
                E[i] = E[j];
                E[j] = temp;
            }
        }
    }
    printf("\n\nThe employee list is successfully sorted by name\n");

}
int delete(struct emp *E, int n)
{
    int i,flag=0;
    char no[10];
    printf("Enter the employee no to be deleted : ");
    scanf("%s",no);
    for(i=0 ; i<n ; i++)
    {
        if (strcmp(E[i].empno,no) == 0)
        {
            flag = 1;
            for( ; i<n-1;i++)
                        {
                                E[i] = E[i+1];
                        }
                        n--;
                        printf("Deleted successfully\n\n\n");
                        break;
                }
    }
    if(flag == 0)
        printf("Employee not found");
```

```
        return n;
}
int menu()
{
        int ch;
        printf("1 . Display Employees\n2 . Search an employee\n3 . Sort by Salary\n4 . Sort
by Name\n5 . Delete an employee\n6 . EXIT\n");
        printf("Enter a choice : ");
        scanf("%d",&ch);
        return ch;
}
void main()
{
   int n;
   struct emp E[20];
   printf("Enter the number of employees : ");
   scanf("%d",&n);
   read(E,n);
   int ch;
        for(ch = menu() ; ch != 6 ; ch = menu())
        {
                switch(ch)
                {
                        case 1 : disp(E,n);
                                        break;
                      case 2 : search(E,n);
                                        break;
                        case 3 : sortSalary(E,n);
                                        break;
                        case 4 : sortName(E,n);
                                        break;
                        case 5 : n = delete(E,n);
                                        break;
```

```
                    default : printf("Wrong choice !!!\n");
                                        break;


            }
            printf("\n");
        }
}
```

Output

```
Enter the number of employees : 3
Enter the details of Employee 1
Enter the Employee No : 101
Enter the name : Anu
Enter the salary : 1500
Enter the department no : 102

Enter the details of Employee 2
Enter the Employee No : 102
Enter the name : Siju
Enter the salary : 2000
Enter the department no : 105

Enter the details of Employee 3
Enter the Employee No : 103
Enter the name : Haritha
Enter the salary : 8000
Enter the department no : 600

1 . Display Employees
2 . Search an employee
3 . Sort by Salary
4 . Sort by Name
5 . Delete an employee
6 . EXIT
Enter a choice : 1
Emp no  Name      Salary        DepNo
101     Anu     1500            102
102     Siju    2000            105
103     Haritha 8000            600


1 . Display Employees
2 . Search an employee
3 . Sort by Salary
4 . Sort by Name
5 . Delete an employee
6 . EXIT
```

```
Enter a choice : 2
Enter employee No to be searched : 101
The employee is found
Emp No : 101
Name : Anu
Salary : 1500
Dep No : 102

1 . Display Employees
2 . Search an employee
3 . Sort by Salary
4 . Sort by Name
5 . Delete an employee
6 . EXIT
Enter a choice : 3


The employee list is successfully sorted by salary

1 . Display Employees
2 . Search an employee
3 . Sort by Salary
4 . Sort by Name
5 . Delete an employee
6 . EXIT
Enter a choice : 4


The employee list is successfully sorted by name

1 . Display Employees
2 . Search an employee
3 . Sort by Salary
4 . Sort by Name
5 . Delete an employee
6 . EXIT
Enter a choice : 5
Enter the employee no to be deleted : 103
Deleted successfully
```

Program#15
Implement a) malloc , b) calloc and c) free functions

Source Code:

```
#include<stdio.h>
#include<malloc.h>
void main()
{
        int *ptr;
        ptr=(int *)malloc(1*sizeof(int));
        printf("Enter a number : ");
        scanf("%d",ptr);
        printf("The value is %d",*ptr);
        int *ptr2;
        ptr2=(int *)calloc(1,sizeof(int));
        printf("\nEnter a number : ");
        scanf("%d",ptr2);
        printf("The value is %d",*ptr2);
        free(ptr);
        free(ptr2);

}
```

Output

```
Enter a number : 5
The value is 5
Enter a number : 10
The value is 10
```

Program#16
Use malloc to read n integers and find the mean.

Source Code:

```
#include<stdio.h>
#include<malloc.h>
void main()
{
        int *ptr,n,sum=0,i;
        float mean;
        printf("Enter value for n : ");
        scanf("%d",&n);
        ptr=(int *)malloc(n*sizeof(int));
        printf("Enter elements\n");
        for(i=0;i<n;i++)
        {
                scanf("%d",ptr+i);
        }
        for(i=0;i<n;i++)
        {
                sum=sum+*(ptr + i);
        }
        mean=sum/n;
        printf("Mean  is %f",mean);
}
```

Output

```
Enter value for n : 4
Enter elements
5
6
4
1
Mean  is 4.000000
```

Program#17
Use calloc to read n numbers and find the mode.

Source Code:

```
#include <stdio.h>
#include <malloc.h>
void read(int *ptr,int n)
{
   int i;
   printf("Enter the elements\n");
   for(i=0;i<n;i++)
   {
      scanf("%d",ptr+i);
   }
}
void mode(int *ptr,int n)
{
  int large = 0, i, j,k=0, count;
  int a[10];
  for (i = 0; i < n; ++i)
  {
    count = 0;
    for (j = i; j < n; ++j)
    {
      if (*(ptr + j) == *(ptr + i))
        ++count;
    }
    if (count >= large)
    {
      if (count > large)
        k=0;
      a[k] = *(ptr + i);
      large = count;
      k++;
    }
```

```
    }
    if(k==1)
        printf("The mode is %d",a[0]);
    else
    {
        printf("The modes are : ");
        for(i=0;i<k;i++)
            printf("%d\t",a[i]);
    }
}
void main()
{
    int i,n;
    printf("Enter the number of elements : ");
    scanf("%d",&n);
    int *ptr = (int *) calloc(n,sizeof(int));
    read(ptr,n);
    mode(ptr,n);
}
```

Output

```
Enter the number of elements : 5
Enter the elements
4
4
2
6
7
The mode is 4
```

Program#18
Declare a structure for Books having author_name and book_name. Create an array of books using a pointer variable. Provide functions for reading n books and displaying the same using pointers.

Source Code:

```
#include<stdio.h>
#include<malloc.h>

struct Book
{
	char author_name[30];
	char book_name[30];
};

void read(struct Book *p, int n)
{
	int i;
	printf("Enter the details\n\n");
	for(i=0;i<n;i++)
	{
		printf("\nBook %d\n",i+1);
		printf("Enter the name of the book : ");//put space before % to prevent skiping
of the input
		scanf(" %[^\n]",(p + i)->book_name);
		printf("Enter the name of the author : ");//put space before % to prevent
skiping of the input
		scanf(" %[^\n]",(p + i)->author_name);
		printf("\n");
	}
}

void display(struct Book *p, int n)
{
```

```
        int i;
        printf("\n\n------Book Details---------\n\n");
        for(i=0;i<n;i++)
        {
                printf("\nBook %d\n",i+1);
                printf("Name : %s\n",(p + i)->book_name);
                printf("Author : %s",(p + i)->author_name);
                printf("\n");
        }
}

void main()
{
        struct Book *p;
        int n;
        printf("Enter the number of books : ");
        scanf("%d",&n);
        p = (struct Book *)malloc(n*sizeof(struct Book));
        read(p,n);
        display(p,n);

}
```

Output

```
Enter the number of books : 3
Enter the details


Book 1
Enter the name of the book : One Indian Girl
Enter the name of the author : Chetan Bhagat


Book 2
Enter the name of the book : A River Sutra
Enter the name of the author : Gita Mehta


Book 3
Enter the name of the book : A changed Man
Enter the name of the author : T.Hardy


------Book Details---------


Book 1
Name : One Indian Girl
Author : Chetan Bhagat

Book 2
Name : A River Sutra
Author : Gita Mehta

Book 3
Name : A changed Man
Author : T.Hardy

--------------------------------
```

Program#19
Use realloc to implement varchar for any length.

Source Code:

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
int main()
{
char*ptr;
char str[50];
int len,n,i;
printf("\nEnter the string : ");
scanf("%s",&str);
len=strlen(str);
ptr=(char*)malloc(len*sizeof(char));
strcpy(ptr,str);
printf("\nThe string using malloc is : ");
for(i=0;i<len;i++)
{
printf("%c",*(ptr+i));
}
printf("\n\nEnter the new size : ");
scanf("%d",&n);
ptr=(char*)realloc(ptr,n);
printf("\nThe string using realloc is : ");
for(i=0;i<n && ptr[i]!='\0';i++)
{
printf("%c",*(ptr+i));
}
free(ptr);
return 0;
}
```

Output

```
Enter the string : malayalam

The string using malloc is : malayalam

Enter the new size : 3

The string using realloc is : mal
```

Program#20
Implement stack operations using array

Source Code:

```
#include<stdio.h>
#define SIZE 5
int stack[SIZE],top=-1;  //global variable
void push()
{
        if(top==SIZE-1)
        {
                printf("Stack is full\n");

        }
        else
        {
                printf("Enter element to be pushed\n");
                top++;  //increment pos by 1(pos=0)
                scanf("%d",&stack[top]);

        }
}
void pop()
{
        if(top==-1)
        {
                printf("Stack is empty\n");
        }
        else
        {
        printf("Popped item is : %d",stack[top]);
        top--;
        }
}
```

```
void peek()
{
	if(top==-1)
	{
		printf("Stack is empty\n");

	}
	else
	{
		printf("Top elements are :%d",stack[top]);
	}
}
int menu()
{
	int ch;
	printf("\n1.Push\n2.Pop\n3.Peek\n4.Exit\nEnter your choice : ");
	scanf("%d",&ch);
	return ch;
}
int main()
{
	int ch;
	ch = menu(); //initialization
	while(ch!=4) //condition
	{
		switch(ch)
		{
			case 1:
				push();
				break;
			case 2:
				pop();
				break;
```

```
                case 3:
                        peek();
                        break;

                default:
                        printf("Wrong choice\n");
                        break;
            }
            ch=menu(); //updation
        }

        return 0;
}
```

Output

```
1.Push
2.Pop
3.Peek
4.Exit
Enter your choice : 1
Enter element to be pushed
10

1.Push
2.Pop
3.Peek
4.Exit
Enter your choice : 3
Top elements are :10
1.Push
2.Pop
3.Peek
4.Exit
Enter your choice : 2
Popped item is : 10
1.Push
2.Pop
3.Peek
4.Exit
```

Program#21
Reverse a string using Stack

Source Code:

```
#include<stdio.h>
#define SIZE 20
char stack[SIZE];
int top = -1;

void push(char item) {
   if(top + 1 == SIZE) {
      printf("ERROR : Stack overflow!! \n");
      return;
   }
   stack[++top] = item;
}
char pop() {
   if(top == -1) {

      printf("ERROR : Stack underflow!! \n");
      return '\0';
   }
   return stack[top--];
}
int main() {

   char ch,n = 0;
   printf("Enter a string (max 30 letters) : ");
   do {
      scanf("%c", &ch);
      push(ch);
      n++;

   }while(ch != '\n');
```

```
    pop();
    printf("Reverse is : ");
    for(n--; n>0; n--) {
        printf("%c", pop());
    }

    return 0;
}
```

Output

```
Enter a string (max 30 letters) : Hindi
Reverse is : idniH
--------------------------------
```

Program#22
Convert an expression from infix to postfix using stack

Source Code:

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include<stdlib.h>
#define MAX_SIZE 100
int isOperator(char ch) {
    return (ch == '+' || ch == '-' || ch == '*' || ch == '/');
}
int precedence(char op) {
    switch (op) {
        case '+':
        case '-':
            return 1;
        case '*':
        case '/':
            return 2;
        default:
            return 0;
    }
}
void infixToPostfix(char infix[], char postfix[]) {
    char opStack[MAX_SIZE];
    int top = -1;
    int i, j;

    for (i = 0, j = 0; infix[i] != '\0'; i++) {
        char ch = infix[i];
        if (ch == ' ')
            continue;
        if (isdigit(ch)) {
            postfix[j++] = ch;
```

```
        } else if (isOperator(ch)) {
            while (top >= 0 && precedence(opStack[top]) >= precedence(ch)) {
                postfix[j++] = opStack[top--];
            }
            opStack[++top] = ch;
        } else if (ch == '(') {
            opStack[++top] = ch;
        } else if (ch == ')') {
            while (top >= 0 && opStack[top] != '(') {
                postfix[j++] = opStack[top--];
            }
            if (top >= 0 && opStack[top] == '(') {
                top--;
            }
        }
        else {
            printf("Invalid token, Quitting");
            exit(0);
        }
    }

    while (top >= 0) {
        postfix[j++] = opStack[top--];
    }

    postfix[j] = '\0';
}

int applyOperator(char op, int operand1, int operand2) {
    switch (op) {
        case '+':
            return operand1 + operand2;
        case '-':
```

```
            return operand1 - operand2;
        case '*':
            return operand1 * operand2;
        case '/':
            return operand1 / operand2;
        default:
            exit(0);
    }
}
int main() {

    char infix[MAX_SIZE];
    char postfix[MAX_SIZE];

    printf("Enter infix string : ");
    scanf("%s", infix);

    infixToPostfix(infix, postfix);

    printf("Postfix expression is : %s\n", postfix);

    return 0;
}
```

Output

```
Enter infix string : 1+2
Postfix expression is : 12+

--------------------------------
```

Program#23
Evaluate an expression using stack

Source Code:

```
#include <stdio.h>
   #include <string.h>
   #include <ctype.h>



   #define MAX_SIZE 100

   int isOperator(char ch) {
      if(ch == '+' || ch == '-' || ch == '*' || ch == '/')
         return 1;
      else
         return 0;
   }

   int applyOperator(char op, int operand1, int operand2) {
   switch (op) {
      case '+':
         return operand1 + operand2;
      case '-':
         return operand1 - operand2;
      case '*':
         return operand1 * operand2;
      case '/':
         return operand1 / operand2;
      }
   }

   void evaluatePostfix(char postfix[]) {


   int i;
   int stack[MAX_SIZE];
```

```
int top = -1;

for(i=0; postfix[i] != '\0'; i++) {

    if(isdigit(postfix[i])) //if digit push it into stack
    {
        stack[++top] = postfix[i] - '0'; //to get the number of the character digit
    }
    else if(isOperator(postfix[i])) {
        int num2 = stack[top--];
        int num1 = stack[top--];
        stack[++top] = applyOperator(postfix[i], num1, num2);
        /*pop two elements from stack.Evaluate it and then push it*/
    }
}
printf("%d", stack[0]);
}
int main()
{
    char str[MAX_SIZE];
    printf("Enter postfix string : ");
    scanf("%s", str);

    evaluatePostfix(str);
}
```

Output

```
Enter postfix string : 23+4-
1
--------------------------------
```

Program#24
A letter means push and an asterisk means pop in the following sequence. Give the sequence of values returned by the pop operations when this sequence of operations is performed on an initially empty LIFO stack. **E A S * Y * Q U E * * * S T * * * I O * N * * ***

Source Code:

```
#include<stdio.h>
#define N 30
char stack[N];
int top=-1;
int push(char e)
{
top=top+1;
stack[top]=e;
return 0;
}
char pop()
{
char ch;
ch=stack[top];
top=top-1;
return ch;
}
int main()
{
int i;
char s;
char str[N];
printf("Enter the string : ");
scanf("%s",str);
while(str[i]!='\0')
{
if(str[i]=='*')
{
```

```
printf("%c",pop());
}
else
{
push(str[i]);
}
i++;
}
return 0;
}
```

Output

```
Enter the string : EAS*Y*QUE***ST***IO*N***
SYEUQTSAONIE
--------------------------------
```

Program#25
Implement Queue using array

Source Code:

```
#include<stdio.h>
int q[5];
int f=-1,r=-1;
void enqueue(int e)
{
if(r+1==5)
{
printf("\nQueue is full");
}
else
{
if(f==-1)
{
f=0;
}
r=r+1;
q[r]=e;
printf("\n enqueued:%d",e);
}
}
void dequeue()
{
int i;
if(f==-1)
{
printf("\nQueue is empty");
}
else
{
printf("\n Dequed element is: %d",q[f]);
if(f==r){
```

```
f=-1;
r=-1;
}
else
{
for(i=0;i<r;i++)
{
q[i]=q[i+1];
}
r=r-1;
}
}
}
int main()
{
int i;
for(i=1;i<5;i++)
{
enqueue(i);
}
for(i=1;i<5;i++)
{
dequeue();
}
}
```

Output

```
enqueued:1
enqueued:2
enqueued:3
enqueued:4
Dequed element is: 1
Dequed element is: 2
Dequed element is: 3
Dequed element is: 4
```

Program#26
Demonstrate circular queue using array.

Source Code:

```
#include<stdio.h>
int f=-1;
int r=-1;
int q[5];
void enqueue(int e)
{
if((r+1)%5==f)
{
printf("\nQueue is full");
}
else
{
if(f==-1)
{
f=0;
}
r=(r+1)%5;
q[r]=e;
printf("\ninserted element is=%d",e);
}
}
void dequeue()
{
if(f==-1)
{
printf("\nQueue empty");
}
else
{
printf("\ndequeued element is=%d",q[f]);
if(f==r)
```

```
{
f=r=-1;
}
else
{
f=(f+1)%5;
}
}
}
int main()
{
enqueue(10);
enqueue(20);
dequeue();
dequeue();
dequeue();
}
```

Output

```
inserted element is=10
inserted element is=20
dequeued element is=10
dequeued element is=20
Queue empty
--------------------------------
```

Program#27
Demonstrate double ended Queue

Source Code:

```
#include <stdio.h>
#define size 5
# define max 100
int deque[size];
int f = -1, r = -1;
void enqueuefront(int x)
{
 if((f==0 && r==size-1) || (f==r+1))
 {
 printf("Overflow");
 }
 else if((f==-1) && (r==-1))
 {
 f=r=0;
 deque[f]=x;
 }
 else if(f==0)
 {
 f=size-1;
 deque[f]=x;
 }
 else
 {
 f=f-1;
 deque[f]=x;
 }
}
void enqueuerear(int x)
{
 if((f==0 && r==size-1) || (f==r+1))
 {
```

```
 printf("Overflow");
 }
 else if((f==-1) && (r==-1))
 {
 r=0;
 deque[r]=x;
 }
 else if(r==size-1)
 {
 r=0;
 deque[r]=x;
 }
 else
 {
 r++;
 deque[r]=x;
 }
}
void display()
{
 int i=f;
 printf("\nElements in a double ended queue are: ");
 while(i!=r)
 {
 printf("%d ",deque[i]);
 i=(i+1)%size;
 }
 printf("%d",deque[r]);
}
void dispfront()
{
 if((f==-1) && (r==-1))
 {
```

```
 printf("Deque is empty");
 }
 else
 {
 printf("\nThe value of the element at front is: %d", deque[f]);
 }
}
void disprear()
{
 if((f==-1) && (r==-1))
 {
 printf("Deque is empty");
 }
 else
 {
 printf("\nThe value of the element at rear is %d", deque[r]);
 }
}
void dequeuefront()
{
 if((f==-1) && (r==-1))
 {
 printf("double ended queue is empty");
 }
 else if(f==r)
 {
 printf("\nThe deleted element is %d", deque[f]);
 f=-1;
 r=-1;
 }
 else if(f==(size-1))
 {
 printf("\nThe deleted element is %d", deque[f]);
```

```
 f=0;
 }
 else
 {
 printf("\nThe deleted element is %d", deque[f]);
 f=f+1;
 }
}
void dequeuerear()
{
 if((f==-1) && (r==-1))
 {
 printf("double ended queue is empty");
 }
 else if(f==r)
 {
 printf("\nThe deleted element is %d", deque[r]);
 f=-1;
 r=-1;
 }
 else if(r==0)
 {
 printf("\nThe deleted element is %d", deque[r]);
 r=size-1;
 }
 else
 {
 printf("\nThe deleted element is %d", deque[r]);
 r=r-1;
 }
}
int main()
{
```

```
int c,a[max];
int x;
while(1)
{
printf("\n1.enqueue rear\n2.enqueue front\n3.dequeue rear\n4.dequeue front\n ENTER
YOURCHOICE\n");
scanf("%d",&c);
switch(c)
{
case 1:
printf("enter the number to enqueue\n");
scanf("%d",&x);
enqueuerear(x);
break;
case 2:
printf("enter the number to enqueue\n");
scanf("%d",&x);
enqueuefront(x);
break;
case 3:
dequeuerear();
break;
case 4:
dequeuefront();
break;
case 5:
break;
}
}
}
```

Output

```
1.enqueue rear
2.enqueue front
3.dequeue rear
4.dequeue front
 ENTER YOURCHOICE
1
enter the number to enqueue
50

1.enqueue rear
2.enqueue front
3.dequeue rear
4.dequeue front
 ENTER YOURCHOICE
2
enter the number to enqueue
6

1.enqueue rear
2.enqueue front
3.dequeue rear
4.dequeue front
 ENTER YOURCHOICE
3

The deleted element is 50
1.enqueue rear
2.enqueue front
3.dequeue rear
4.dequeue front
 ENTER YOURCHOICE
4

The deleted element is 6
```

Program#28
Implement priority queue

Source Code:

```
#include <stdio.h>
int size = 0;
void swap(int *a, int *b)
{
 int temp = *b;
 *b = *a;
*a = temp;
}
void heapify(int array[], int size, int i)
{
int l,r,largest;
if (size == 1)
{
printf("Single element in the heap");
}
else
{
 largest = i;
 l = 2 * i + 1;
 r = 2 * i + 2;
 if (l < size && array[l] > array[largest])
 largest = l;
 if (r < size && array[r] > array[largest])
 largest = r;
 if (largest != i)
{
 swap(&array[i], &array[largest]);
 heapify(array, size, largest);
 }
 }
}
```

```
void insert(int array[], int newNum)
{
int i;
 if (size == 0)
{
 array[0] = newNum;
 size += 1;
}
else
{
 array[size] = newNum;
 size += 1;
 for (i = size / 2 - 1; i >= 0; i--)
{
 heapify(array, size, i);
 }
 }
}
void deleteRoot(int array[], int num)
{
 int i;
 for (i = 0; i < size; i++)
{
 if (num == array[i])
 break;
 }
 swap(&array[i], &array[size - 1]);
 size -= 1;
 for (i = size / 2 - 1; i >= 0; i--)
{
 heapify(array, size, i);
 }
}
```

```
void printArray(int array[], int size)
{
int i;
 for (i = 0; i < size; ++i)
 printf("%d ", array[i]);
 printf("\n");
}
int main() {
 int array[10];
 insert(array, 3);
 insert(array, 4);
 insert(array, 9);
 insert(array, 5);
 insert(array, 2);
 printf("Max-Heap array: ");
 printArray(array, size);
 deleteRoot(array, 4);
 printf("After deleting an element: ");
 printArray(array, size);
}
```

Output

```
Max-Heap array: 9 5 4 3 2
After deleting an element: 9 5 2 3

--------------------------------
```

Program#29
Implement heapsort

Source Code:

```
#include <stdio.h>
#define SIZE 5
void swap(int *a, int *b)
{
int temp = *a;
*a = *b;
*b = temp;
}
void heapify(int arr[], int n, int i)
{
int largest = i;
int left = 2 * i + 1;
int right = 2 * i + 2;
if (left < n && arr[left] > arr[largest])
largest = left;
if (right < n && arr[right] > arr[largest])
largest = right;
if (largest != i)
{
swap(&arr[i], &arr[largest]);
heapify(arr, n, largest);
}
}
void heapSort(int arr[], int n)
{
int i;
for ( i = n / 2 - 1; i >= 0; i--)
heapify(arr, n, i);
for ( i = n - 1; i >= 0; i--){
swap(&arr[0], &arr[i]);
heapify(arr, i, 0);}}
```

```
void printArray(int arr[], int n)
{
int i;
for ( i = 0; i < n; ++i)
printf("%d ", arr[i]);
printf("\n");
}
int main()
{
int i;
int arr[SIZE];
printf("\nArray Size : %d\n",SIZE);
for( i=0; i<SIZE; i++)
{
printf("\nEnter the array [%d] element --> ",i+1);
scanf("%d",&arr[i]);
}
int n = sizeof(arr) / sizeof(arr[0]);
heapSort(arr, n);
printf("\nHeap sorted array is : ");
printArray(arr, n);
return 0;
}
```

Output

```
Array Size : 5
Enter the array [1] element --> 20
Enter the array [2] element --> 10
Enter the array [3] element --> 8
Enter the array [4] element --> 6
Enter the array [5] element --> 4
Heap sorted array is : 4 6 8 10 20
```

Program#30
Search an element using binary search

Source Code:

```
#include <stdio.h>
int main()
{
int i, low, high, mid, n, key, array[100];
printf("Enter number of elements\n");
scanf("%d",&n);
printf("Enter %d integers\n", n);
for(i = 0; i < n; i++)
scanf("%d",&array[i]);
printf("Enter value to find\n");
scanf("%d", &key);
low = 0;
high = n - 1;
mid = (low+high)/2;
while (low <= high) {
if(array[mid] < key)
low = mid + 1;
else if (array[mid] == key) {
printf("%d found at location %d\n", key, mid+1);
break;
}
else
high = mid - 1;
mid = (low + high)/2;
}
if(low > high)
printf("Not found! %d isn't present in the list\n", key);
return 0;
}
```

Output

```
Enter number of elements
5
Enter 5 integers
2
8
9
10
30
Enter value to find
9
9 found at location 3
```

Program#31

Implement various sorting Algorithms

a. **Bubble Sort**
b. **Selection Sort**
c. **Insertion Sort**

Source Code:

```
#include <stdio.h>
int readArray(int a[]) {
int n,i;
printf("Enter the no of elements");
scanf("%d",&n);
for(i = 0;i < n;i++) {
printf("Enter a[%d]",i);
scanf("%d",&a[i]);
}
return n;
}
void dispArray(int a[],int n) {
int i;
printf("\n");
for(i = 0;i < n;i++) {
printf("%d\t",a[i]);
}
}
void insertionSort(int a[],int n) {
int i,j,e,t;
for(i = 1;i < n;i++) {
e = a[i];
j = i-1;
while(j>=0 && a[j] > e) {
a[j+1] = a[j];
j--;
}
```

```
a[j+1] = e;
}
}
void selectionSort(int a[],int n) {
int i,j,t;
for(i = 0;i < n;i++) {
for(j = i+1;j < n;j++) { // i = 0 j from 1 to n-1
if(a[i] > a[j]) { // i = 1 j from 2 to n-1
t = a[i];
a[i] = a[j];
a[j] = t;
}
}
}
}
void bubbleSort(int a[],int n) {
int i,j,t;
for(i = 0;i < n;i++) {
for(j = 0; j < n-i-1;j++) {
if(a[j] > a[j+1]) {
t = a[j];
a[j] = a[j + 1];
a[j + 1] = t;
}
}
}
}
int main()
{
int a[10],n,ch;
while(1)
{
printf("\n1.BUBBLE SORT\n2.SELECTION SORT\n3.INSERTION SORT\n4.EXIT\n");
```

```
printf("\nenter your choice\n");
scanf("%d",&ch);
switch(ch)
{
case 1:
n = readArray(a);
dispArray(a,n);
bubbleSort(a,n);
printf("\nafter bubble sorting :\n");
dispArray(a,n);
break;
case 2:
n = readArray(a);
dispArray(a,n);
selectionSort(a,n);
printf("\nafter selection sorting :\n");
dispArray(a,n);
break;
case 3:
n = readArray(a);
dispArray(a,n);
insertionSort(a,n);
printf("\nafter insertion sorting :\n");
dispArray(a,n);
break;
case 4:
break;
default:
printf("wrong choice");
}
}
}
```

Output

```
1.BUBBLE SORT
2.SELECTION SORT
3.INSERTION SORT
4.EXIT

enter your choice
1
Enter the no of elements3
Enter a[0]5
Enter a[1]9
Enter a[2]4

5       9       4
after bubble sorting :

4       5       9
1.BUBBLE SORT
2.SELECTION SORT
3.INSERTION SORT
4.EXIT

enter your choice
2
Enter the no of elements3
Enter a[0]5
Enter a[1]9
Enter a[2]3

5       9       3
after selection sorting :

3       5       9
1.BUBBLE SORT
2.SELECTION SORT
3.INSERTION SORT
4.EXIT

enter your choice
3
```

```
enter your choice
3
Enter the no of elements3
Enter a[0]5
Enter a[1]9
Enter a[2]3

5       9       3
after insertion sorting :

3       5       9
1.BUBBLE SORT
2.SELECTION SORT
3.INSERTION SORT
4.EXIT
```

Program#32
Demonstrate the sparse representation for a given matrix

Source Code:

```
#include<stdio.h>
void main()
{
int a[10][10];
int r,c,i,j;
printf("enter your row size\n");
scanf("%d",&r);
 printf("enter the column size\n");
 scanf("%d",&c);
 printf("enter th sparese matrix\n");
 for(i=0;i<r;i++)
 {
 for(j=0;j<c;j++)
 {
 scanf("%d",&a[i][j]);
}
 }
 printf("your matrix is :\n");
 for(i=0;i<r;i++)
 {
 for(j=0;j<c;j++)
 {
 printf("%d\t",a[i][j]);
}
printf("\n");
}
int s=0;
 int b[10][3];
for(i=0;i<r;i++)
{
for(j=0;j<c;j++)
```

```
{
if(a[i][j]!=0)
{
b[s][0]=i;
b[s][1]=j;
b[s][2]=a[i][j];
s++;}
}
}
printf("the sparse matrix representation is: \n");
for(i=0;i<s;i++)
{
for(j=0;j<3;j++)
{
printf("%d\t",b[i][j]);
}
printf("\n");
}
}
```

Output

```
enter your row size
3
enter the column size
2
enter th sparese matrix
3
6
5
4
2
8
your matrix is :
3       6
5       4
2       8
the sparse matrix representation is:
0       0       3
0       1       6
1       0       5
1       1       4
2       0       2
2       1       8
```

Program#33
Demonstrate Circular Linked List

Source Code:

```
#include <stdio.h>
#include <malloc.h>
struct node
{
   int data;
   struct node *next;
};
struct node *head = NULL;
void insert(int e)
{struct node *t;
   int a;
   if(head == NULL)
   {
      head = (struct node *)malloc(sizeof(struct node));
      head->data = e;
      head->next = head;
   }
   else
   {  t = head;
      while(t->next != head)
      {
         t = t->next;
      }
      t->next = (struct node *)malloc(sizeof(struct node));
      t->next->data = e;
      t->next->next = head;
   }
}
void delete(int e)
{  struct node *t;
   if(head == NULL)
```

```
    {
        printf("\nLinked List is empty!!!\n\n");
    }
    else  if(head->data == e & head->next == head)
    {
        head=NULL;
    }
    else if(head->data == e)
    {
            t=head;
            while(t->next!=head){
                    t =t->next;
                    }
            t->next = head->next;
            head = head->next;
            }
            else
    {
        t=head;
        while(t->next != head && t->next->data != e)
        {
            t=t->next;
        }
        if(t->next == head)
            printf("\nElement not found\n\n");
        else
        {
            t->next=t->next->next;
        }
    }
}
void disp()
{   struct node *t;
```

```
    if(head == NULL)
    {
        printf("\nLinked List is empty!!!\n\n");
    }
    else
    {
        t=head;
        printf("\n");
        do
        {
            printf("%d\t",t->data);
            t = t->next;
        }
            while(t!=head);
        printf("\n\n");
    }
}
int menu()
{  int ch;
    printf("\n1 - Insert\n2 - Delete\n3 - Display\n4 - Exit\n");
    printf("Enter your choice : ");
    scanf("%d",&ch);
    return ch;
}
int main()
{ int i,ch,a;
    for(ch = menu();ch != 4;ch = menu())
    {
        switch(ch)
        {
            case 1 : printf("Enter an element to insert : ");
                                scanf("%d",&a);
                                insert(a);
```

```
                    break;
        case 2 : printf("Enter an element to delete : ");
                    scanf("%d",&a);
                    delete(a);
                     break;
        case 3 : disp();
                break;
        default : printf("Wrong Choice!!!\n");
                break;
    };
  }
  return 0;
}
```

Output

```
1 - Insert
2 - Delete
3 - Display
4 - Exit
Enter your choice : 1
Enter an element to insert : 2

1 - Insert
2 - Delete
3 - Display
4 - Exit
Enter your choice : 1
Enter an element to insert : 10

1 - Insert
2 - Delete
3 - Display
4 - Exit
Enter your choice : 2
Enter an element to delete : 10

1 - Insert
2 - Delete
3 - Display
4 - Exit
Enter your choice : 3

2
```

Program#34
Demonstrate Doubly Linked List

Source Code:

```
#include<stdio.h>
#include<malloc.h>
struct node
{
        int data;
        struct node *next;
        struct node *prev;
};
struct node *head = NULL;
void insert(int e)
{       struct node *t;
        if(head==NULL)
        {
                head = (struct node *)malloc(sizeof(struct node));
                head->data = e;
                head->prev = head->next = NULL;
        }
        else
        {t=head;
                while(t->next != NULL)
                        t=t->next;
                t->next = (struct node *)malloc(sizeof(struct node));
                t->next->data = e;
                t->next->prev = t;
                t->next->next = NULL;
        }
}
void delete(int e)
{struct node *t;
        if(head==NULL)
        {
```

```
            printf("Empty!");
        }
        else if(head->data ==e && head->next == NULL)
        {
            head = NULL;
        }
        else if(head->data == e)
        {
            head=head->next;
            head->prev = NULL;
        }
        else
        {t=head;
            while(t!=NULL && t->data!=e)
                t=t->next;
            if(t==NULL)
                printf("Element not found\n");
            else if(t->next == NULL)
            {
                t->prev->next = NULL;
            }
            else
            {
                t->prev->next = t->next;
                t->next->prev = t->prev;
            }
        }
}

void display()
{struct node *t;
        if(head == NULL)
        {
```

```
                printf("Doubly Linked List empty\n");
        }
        else
        {t=head;
                while(t!=NULL)
                {
                        printf("%d\t",t->data);
                        t=t->next;

                }
                printf("\n");
        }
}
int menu()
{  int ch;
   printf("\nInsert - 1\nDelete - 2\nDisplay - 3\nExit - 4\n");
   printf("Enter your choice : ");
   scanf("%d",&ch);
   return ch;
}
int main()
{int i,ch,e;
   for(ch = menu();ch != 4;ch = menu())
   {
      switch(ch)
      {
         case 1 : printf("Enter the value to be inserted : ");
                        scanf("%d",&e);
                        insert(e);
                        break;
         case 2 : printf("Enter the value to be deleted : ");
                        scanf("%d",&e);
                        delete(e);
```

```
                    break;
        case 3 : display();
                break;
        default : printf("Wrong Choice!!!\n");
                break;
    };
  }
  return 0;
}
```

Output

```
Insert - 1
Delete - 2
Display - 3
Exit - 4
Enter your choice : 1
Enter the value to be inserted : 10

Insert - 1
Delete - 2
Display - 3
Exit - 4
Enter your choice : 1
Enter the value to be inserted : 20

Insert - 1
Delete - 2
Display - 3
Exit - 4
Enter your choice : 1
Enter the value to be inserted : 30

Insert - 1
Delete - 2
Display - 3
Exit - 4
Enter your choice : 2
Enter the value to be deleted : 20

Insert - 1
Delete - 2
Display - 3
Exit - 4
Enter your choice : 3
10      30
```

Program#35
Demonstrate Circular doubly linked list

Source Code:

```
#include <stdio.h>
#include <stdlib.h>
struct node {
int data;
struct node *next;
struct node *prev;
};
typedef struct node cdlist;
cdlist *head = NULL;
void insert(int e) {
cdlist *t;
if(head == NULL){
head = (cdlist *)malloc(sizeof(cdlist));
head->data = e;
head->next = head;
head->prev = head;
}
else {
t = head;
while(t->next != head) {
t = t->next;
}
t->next = (cdlist *)malloc(sizeof(cdlist));
t->next->data = e;
t->next->next = head;
t->next->prev = t;
head->prev = t->next;
}
}
void disp() {
cdlist *t;
```

```
if(head == NULL){
printf("DList is empty");
}
else {
t = head;
do {
printf("%d\t",t->data);
t = t->next;
}while(t != head);
printf("\n");
}
}
void Delete(int e) {
cdlist *t;
if(head == NULL){ // list is empty
printf("CDList is empty");
}
else if(head->data == e && head->next == head){ // first element with no element or several
elements
 head = NULL;
}
else if(head->data == e) {
 head->next->prev = head->prev;
 head->prev->next = head->next;
 head = head->next;
}
else {
t = head->next;
while(t != head && t->data != e){
t = t->next;
}
if(t == head) {
printf("Not Found");
```

```
}
else { //intermediate
t->prev->next = t->next;
t->next->prev = t->prev;
}
}
}
int menu() {
int ch;
printf("1. Insert\n2. Display\n3. Delete\n4.exit\nEnter your Choice");
scanf("%d",&ch);
return ch;
}
void processList() {
int ch,nv;
for(ch = menu();ch != 4;ch = menu()) {
switch(ch) {
case 1:
printf("Enter the element");
scanf("%d",&ch);
insert(ch);
break;
case 2:
disp();
break;
case 3:
printf("Enter the element");
scanf("%d",&ch);
Delete(ch);
break;
case 4:
break;
default:
```

```
printf("Wrong Choice");
break;
}
}
}
int main() {
processList();
return 0;
}
```

Output

```
1. Insert
2. Display
3. Delete
4.exit
Enter your Choice1
Enter the element5
1. Insert
2. Display
3. Delete
4.exit
Enter your Choice1
Enter the element60
1. Insert
2. Display
3. Delete
4.exit
Enter your Choice1
Enter the element70
1. Insert
2. Display
3. Delete
4.exit
Enter your Choice3
Enter the element70
1. Insert
2. Display
3. Delete
4.exit
Enter your Choice2
5       60
1. Insert
2. Display
3. Delete
4.exit
Enter your Choice4

--------------------------------
```

Program#36
Merge two sorted linked lists to a single sorted linked list. Do not sort after combining both lists.

Source Code:

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
 int info;
 struct node *link;
};
struct node *create(struct node *start);
struct node *insert_s(struct node *start,int data);
struct node *insert(struct node *start,int data);
void display(struct node *start );
void merge(struct node *p1,struct node *p2);
int main()
{
 struct node *start1=NULL,*start2=NULL;
 start1=create(start1);
 start2=create(start2);
 printf("List1 : ");
 display(start1);
 printf("List2 : ");
 display(start2);
 merge(start1, start2);
 return 0;
}/*End of main()*/
void merge(struct node *p1,struct node *p2)
{
 struct node *start3;
 start3=NULL;
 while(p1!=NULL && p2!=NULL)
 {
```

```
if(p1->info < p2->info)
{
start3=insert(start3,p1->info);
p1=p1->link;
}
else if(p2->info < p1->info)
{
start3=insert(start3,p2->info);
p2=p2->link;
}
else if(p1->info==p2->info)
{
start3=insert(start3,p1->info);
p1=p1->link;
p2=p2->link;
}
}
while(p1!=NULL)
{
start3=insert(start3,p1->info);
p1=p1->link;
}
while(p2!=NULL)
{
start3=insert(start3,p2->info);
p2=p2->link;
}
printf("Merged list is : ");
display(start3);
}
struct node *create(struct node *start )
{
int i,n,data;
```

```
 printf("Enter the number of nodes : ");
 scanf("%d",&n);
 start=NULL;
 for(i=1;i<=n;i++)
 {
 printf("Enter the element to be inserted : ");
 scanf("%d",&data);
 start=insert_s(start, data);
 }
 return start;
}
struct node *insert_s(struct node *start,int data)
{
 struct node *p,*tmp;
 tmp=(struct node *)malloc(sizeof(struct node));
 tmp->info=data;
 if(start==NULL || data<start->info)
 {
 tmp->link=start;
 start=tmp;
 return start;
 }
 else
 {
 p=start;
 while(p->link!=NULL && p->link->info < data)
 p=p->link;
 tmp->link=p->link;
 p->link=tmp;
 }
 return start;
}/*End of insert_s()*/
struct node *insert(struct node *start,int data)
```

```
{
 struct node *p,*tmp;
 tmp=(struct node *)malloc(sizeof(struct node));
 tmp->info=data;
 /*If list is empty*/
 if(start==NULL)
 {
 tmp->link=start;
 start=tmp;
 return start;
 }
 else /*Insert at the end of the list*/
 {
 p=start;
 while(p->link!=NULL)
 p=p->link;
 tmp->link=p->link;
 p->link=tmp;
 }
 return start;
}
void display(struct node *start)
{
 struct node *p;
 if(start==NULL)
 {
 printf("List is empty\n");
 return;
 }
 p=start;
 while(p!=NULL)
 {
 printf("%d ",p->info);
```

```
 p=p->link;
 }
 printf("\n");
}
```

Output

```
Enter the number of nodes : 4
Enter the element to be inserted : 5
Enter the element to be inserted : 7
Enter the element to be inserted : 8
Enter the element to be inserted : 10
Enter the number of nodes : 4
Enter the element to be inserted : 60
Enter the element to be inserted : 1
Enter the element to be inserted : 20
Enter the element to be inserted : 33
List1 : 5 7 8 10
List2 : 1 20 33 60
Merged list is : 1 5 7 8 10 20 33 60
```

Program#37
Linked Stack

Source Code:

```
#include<stdio.h>
#include<malloc.h>

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

typedef struct node stack;

stack *top=NULL;
void push(int e)
{
	stack *t = (stack *)malloc(sizeof(stack));
	t->data=e;
	t->next=top;
	top=t;
}

void pop()
{
	if(top==NULL)
		printf("Stack Underflow\n");
	else
	{
		printf("%d\n",top->data);
		top = top->next;
	}
}
```

```
void peek()
{
        if(top==NULL)
                printf("Empty Stack\n");
        else
                printf("%d\n",top->data);
}


int menu()
{
   int ch;
   printf("\nPush - 1\nPop - 2\nPeek - 3\nExit - 4\n");
   printf("Enter your choice : ");
   scanf("%d",&ch);
   return ch;
}

void main()
{
   int i,ch,a;
   for(ch = menu();ch != 4;ch = menu())
   {
      switch(ch)
      {
         case 1 : printf("Enter an element to insert : ");
                        scanf("%d",&a);
                                push(a);
              break;
         case 2 : pop();
              break;
         case 3 : peek();
```

```
                break;
        default : printf("Wrong Choice!!!\n");
                break;
    };
  }
}
```

Output

```
Push - 1
Pop - 2
Peek - 3
Exit - 4
Enter your choice : 1
Enter an element to insert : 5

Push - 1
Pop - 2
Peek - 3
Exit - 4
Enter your choice : 1
Enter an element to insert : 20

Push - 1
Pop - 2
Peek - 3
Exit - 4
Enter your choice : 1
Enter an element to insert : 30

Push - 1
Pop - 2
Peek - 3
Exit - 4
Enter your choice : 2
30

Push - 1
Pop - 2
Peek - 3
Exit - 4
Enter your choice : 3
20
```

Program#38
Reverse a name using a linked stack.

Source Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define M 100
struct Stack{
char ele;
struct Stack *next;
};
struct Stack* next_node(char element)
{
struct Stack *node=(struct Stack *)malloc(sizeof(struct Stack)); node->ele=element;
node->next=NULL;
return node;
}
int isEmpty(struct Stack *node)
{
return node==NULL;
}
void push(struct Stack **node, char element)
{
struct Stack *temp=next_node(element);
temp->next=*node;
*node=temp;
}
char pop(struct Stack** node)
{
if (isEmpty(*node))
{
return 0;
}
```

```
struct Stack* temp = *node;
*node = (*node)->next;
char retval = temp->ele;
free(temp);
return retval;
}
void rev(char str[])
{
int i;
int n = strlen(str);
struct Stack* s = NULL;
for (i = 0; i < n; i++)
push(&s, str[i]);
for (i = 0; i < n; i++)
str[i] = pop(&s);
printf("The reversed string is: %s\n", str);
}
int main()
{
char string[M], op[1];
printf("Enter the string to be reversed: ");
scanf("%s", string);
rev(string);
return 0;
}
```

Output

```
Enter the string to be reversed: Data
The reversed string is: ataD

--------------------------------
```

Program#39
Linked Queue

Source Code:

```
#include<stdio.h>
#include<malloc.h>
struct node
{
	int data;
	struct node *next;
};
typedef struct node queue;
queue *front=NULL;
queue *rear=NULL;
void enqueue(int e)
{queue *t = (queue *)malloc(sizeof(queue));
	t->data=e;
	t->next=NULL;
	if(front == NULL)
	{
		front = t;
		rear = t;
	}
	else
	{
		rear->next = t;
		rear = t;
	}
}
void dequeue()
{	if(front==NULL)
		printf("Queue Empty\n");
	else
	{printf("%d\n",front->data);
```

```
                front = front->next;
                if(front == NULL)
                        rear=NULL;
        }
}
void display()
{if(front==NULL)
                printf("Empty Queue\n");
        else
        {
                queue *t=front;
                while(t != NULL)
                {
                        printf("%d\t",t->data);
                        t=t->next;
                }
        }
}
int menu()
{ int ch;
   printf("\nEnqueue - 1\nDequeue - 2\nDisplay - 3\nExit - 4\n");
   printf("Enter your choice : ");
   scanf("%d",&ch);
   return ch;
}
void main()
{   int i,ch,a;
   for(ch = menu();ch != 4;ch = menu())
   {switch(ch)
      {
         case 1 : printf("Enter an element to insert : ");
                        scanf("%d",&a);
                        enqueue(a);
```

```
                    break;
        case 2 : dequeue();
                break;
        case 3 : display();
                break;
        default : printf("Wrong Choice!!!\n");
                break;
    };
  }
}
```

Output

```
Enqueue - 1
Dequeue - 2
Display - 3
Exit - 4
Enter your choice : 1
Enter an element to insert : 10

Enqueue - 1
Dequeue - 2
Display - 3
Exit - 4
Enter your choice : 1
Enter an element to insert : 20

Enqueue - 1
Dequeue - 2
Display - 3
Exit - 4
Enter your choice : 1
Enter an element to insert : 30

Enqueue - 1
Dequeue - 2
Display - 3
Exit - 4
Enter your choice : 2
10

Enqueue - 1
Dequeue - 2
Display - 3
Exit - 4
Enter your choice : 3
20      30
```

Program#40
Write a program to sort a linked list of names using bubble sort.

Source Code:

```
#include<stdio.h>
#include<stdlib.h>
#include <string.h>
typedef struct nodes
{
char data[20];
struct nodes *next;
}node;
node *head;
node *getnode()
{
node *ptr;
char c[20];
ptr=(node *)malloc(sizeof(node));
printf("Enter the name ");
scanf("%s",&c);
strcpy(ptr->data,c);
return ptr;
}
void sortlist(int n);
void printList();
void insertion()
{
node *ptr,*temp;
ptr=getnode();
if(head==NULL)
{
head=ptr;
ptr->next=NULL;
}
else
```

```
{
temp=head;
while(temp->next!=NULL) temp=temp->next;
temp->next=ptr; ptr->next=NULL;
}
}
int main()
{
head=NULL;
int n,i;
printf("Enter the number of names to be added : ");
scanf("%d",&n);
for(i=0;i<n;i++)
{
insertion();
}
printf("\n Linked list before sorting ");
printList();
sortlist(n);
printf("\n Linked list after sorting ");
printList();
return 0;
}
void sortlist(int n)
{
char t[25];
int i,j;
node *temp;
for(i=0;i<n;i++)
{
temp=head;
while(temp->next!=NULL)
{
```

```
if(strcmp(temp->data,temp->next->data)>0)
{
strcpy(t,temp->next->data); strcpy(temp->next->data,temp->data);
strcpy(temp->data,t);
}
temp=temp->next;
}
}
}
void printList()
{
node *temp=head;
printf("\n");
while (temp!=NULL)
{
printf("%s,\t", temp->data);
temp = temp->next;
}
}
```

Output

```
Enter the number of names to be added : 3
Enter the name Alfard
Enter the name Sophy
Enter the name Bella

 Linked list before sorting
Alfard, Sophy,  Bella,
 Linked list after sorting
Alfard, Bella,  Sophy,
--------------------------------
```

Program#41
Demonstrate Circular Linked Queue

Source Code:

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
int data;
struct node *next;
};
typedef struct node queue;
queue *front=NULL;
queue *rear=NULL;
void enqueue()
{queue *t=(queue*)malloc(sizeof(queue));
printf("\nEnter the element to be inserted : ");
scanf("%d",&t->data);
if(front==NULL)
front=t;
if(rear==NULL)
rear=t;
else
{
rear->next=t;
rear=rear->next;
}
rear->next=front;
}
void dequeue()
{if(front==NULL)
printf("\nCircular Linked Queue is Empty");
else
{
```

```
{
}
}
printf("\nDequeued element : %d",front->data);
if(front==rear)
front=rear=NULL;
rear->next=front->next;
front=front->next;
}
void display()
{printf("\n");
queue *t=front;
do
{printf("%d ",t->data);
t=t->next;
}while(t!=front);
}
int main()
{int ch;
do
{
printf("\n");
printf("\n*** MENU ***\n");
printf("\n1. Insertion");
printf("\n2. Deletion");
printf("\n3. Display");
printf("\n4. Exit");
printf("\nEnter your choice (1-4) : ");
scanf("%d",&ch);
switch(ch)
{
case 1 : enqueue();
break;
```

```
case 2 : dequeue();
break;
case 3 : display();
break;
case 4 : break;
default : printf("\nInvalid Choice\n");
}
}while(ch!=4);
return 0;
}
```

Output

```
1. Insertion
2. Deletion
3. Display
4. Exit
Enter your choice (1-4) : 1

Enter the element to be inserted : 10


*** MENU ***

1. Insertion
2. Deletion
3. Display
4. Exit
Enter your choice (1-4) : 1

Enter the element to be inserted : 20


*** MENU ***

1. Insertion
2. Deletion
3. Display
4. Exit
Enter your choice (1-4) : 2

Dequeued element : 10

*** MENU ***

1. Insertion
2. Deletion
3. Display
4. Exit
Enter your choice (1-4) : 3

20
```

Program#42
Binary search tree insertion and implement Traversal using inorder, preorder and postorder using recursion

Source Code:

```
#include <stdio.h>
#include <stdlib.h>
struct node {
int item;
struct node* left;
struct node* right;
};
void inorderTraversal(struct node* root)
{
if (root == NULL)
return;
inorderTraversal(root->left);
printf("%d ->", root->item);
inorderTraversal(root->right);
}
void preorderTraversal(struct node* root)
{
if (root == NULL)
return;
printf("%d ->", root->item);
preorderTraversal(root->left);
preorderTraversal(root->right);
}
void postorderTraversal(struct node* root)
{
if (root == NULL)
return;
postorderTraversal(root->left);
postorderTraversal(root->right);
printf("%d ->", root->item);
```

```
}
struct node* createNode(value)
{
struct node* newNode = malloc(sizeof(struct node));
newNode->item = value;
newNode->left = NULL;
newNode->right = NULL;
return newNode;
}
struct node* insertLeft(struct node* root, int value)
{
root->left = createNode(value);
return root->left;
}
struct node* insertRight(struct node* root, int value)
{
root->right = createNode(value);
return root->right;
}
int main()
{
int a,b,c,d,e;
printf("Enter the rootnode");
scanf("%d",&a);
struct node* root = createNode(a);
printf("Enter the node"); scanf("%d",&b);
insertLeft(root, b);
printf("Enter the node");
scanf("%d",&c);
insertRight(root, c);
printf("Enter the node");
scanf("%d",&d); insertLeft(root->left, d);
printf("Enter the node");
```

```
scanf("%d",&e);
insertRight(root->left, e);
printf("Inorder traversal \n");
inorderTraversal(root);
printf("\nPreorder traversal \n");
preorderTraversal(root);
printf("\nPostorder traversal \n");
postorderTraversal(root);
}
```

Output

```
Enter the rootnode10
Enter the node3
Enter the node2
Enter the node5
Enter the node2
Inorder traversal
5 ->3 ->2 ->10 ->2 ->
Preorder traversal
10 ->3 ->5 ->2 ->2 ->
Postorder traversal
5 ->2 ->3 ->2 ->10 ->
--------------------------------
```

Program#43
Binary search tree insertion and implement Traversal using inorder, preorder and postorder without using recursion

Source Code:

```
#include <stdio.h>
#include <stdlib.h>
typedef struct node
{
 int data;
 struct node *left;
 struct node *right;
} Node;
typedef struct stack
{
 Node *node;
 struct stack *next;
} LStack;
typedef struct poststack
{
 Node *node;
 int count;
 struct poststack *next;
} Poststack;
LStack *top = NULL;
LStack *head = NULL;
Poststack *ptop = NULL;
Poststack *phead = NULL;
Node *tree = NULL;
void insert(int data)
{
 Node *new_node = (Node *)malloc(sizeof(Node));
 new_node->data = data;
 new_node->left = NULL;
 new_node->right = NULL;
```

```
if (tree == NULL)
{
tree = new_node;
}
else
{
Node *t = tree, *x = NULL;
while (t != NULL)
{
x = t;
if (data < t->data)
{
t = t->left;
}
else
{
t = t->right;
}
}
if (data < x->data)
{
x->left = new_node;
}
else
{
x->right = new_node;
}
}
}
void push(Node *t)
{
LStack *new_node = (LStack *)malloc(sizeof(LStack));
new_node->node = t;
```

```
 if (head == NULL)
 {
 head = new_node;
 head->next = top;
 top = head;
 }
 else
 {
 new_node->next = top;
 top = new_node;
 }
}
Node *pop()
{
 if (top != NULL)
 {
 Node *x = top->node;
 top = top->next;
 return x;
 }
 else
 {
 return NULL;
 }
}
void postStackPush(Node *t)
{
 Poststack *new_node = (Poststack *)malloc(sizeof(Poststack));
 new_node->node = t;
 new_node->count = 1;
 if (phead == NULL)
 {
 new_node->next = ptop;
```

```
 phead = new_node;
 ptop = phead;
 }
 else
 {
 new_node->next = ptop;
 ptop = new_node;
 }
}
Node *postStackPop()
{
 if (ptop != NULL)
 {
 Node *x = ptop->node;
 ptop = ptop->next;
 return x;
 }
 else
 {
 return NULL;
 }
}
void postOrder()
{
 if (tree == NULL)
 {
 return;
 }
 Node *current = tree;
 while (current != NULL)
 {
 postStackPush(current);
 current = current->left;
```

```
 }
 Poststack *i = ptop;
 Node *c;
 for (i = ptop; i != NULL; i = ptop)
 {
 if (i->count == 2)
 {
 c = postStackPop();
 printf("%d\t", c->data);
 }
 else
 {
 i->count = 2;
 if (i->node->right != NULL)
 {
 current = i->node->right;
 while (current != NULL)
 {
 postStackPush(current);
 current = current->left;
 }
 }
 }
 }
}
void inorder()
{
 if (tree == NULL)
 {
 return;
 }
 Node *current = tree;
 while (current != NULL)
```

```
{
push(current);
current = current->left;
}
Node *i;
for (i= pop();i!= NULL; i = pop())
{
current = i;
printf("%d\t", current->data);
if (current->right != NULL)
{
current = current->right;
while (current != NULL)
{
push(current);
current = current->left;
}
}
}
printf("\n");
}
void preorder()
{
// Print and then Push to the Stack
if (tree == NULL)
{
return;
}
Node *current = tree, *i;
while (current != NULL)
{
printf("%d\t", current->data);
push(current);
```

```
current = current->left;
}
// Now Pop and check the right of the Stack
for (i = pop(); i != NULL; i = pop())
{
current = i;
if (current->right != NULL)
{
current = current->right;
while (current != NULL)
{
printf("%d\t", current->data);
push(current);
current = current->left;
}
}
}
printf("\n");
}
void main()
{
insert(20);
insert(10);
insert(5);
insert(100);
insert(50);
insert(150);
insert(6);
insert(13);
printf("Inorder\n");
inorder();
printf("Preorder\n");
preorder();
```

```
 printf("Postorder\n");
 postOrder();
 free(head);
 free(top);
 free(tree);
}
```

Output

```
Inorder
5       6       10      13      20      50      100     150
Preorder
20      10      5       6       13      100     50      150
Postorder
6       5       13      10      50      150     100     20
--------------------------------
```

Program#44
Binary search tree insertion using names and display the names in ascending order using inorder traversal.

Source Code:

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
struct node {
char name[20];
struct node *left, *right;
};
typedef struct node tree;
tree *root=NULL;
void insert(char e[20])
{
tree *t, *x;
if(root==NULL)
{
root= (tree*)malloc(sizeof(tree));
strcpy(root->name,e);
root->left=NULL;
root->right=NULL;
}
else
{
t=root;
while(t != NULL)
{
x=t;
if(strcmp(t->name,e) == 0)
{
printf("/n Duplicate name");
return;
}
```

```
else if(strcmp(t->name,e) > 0)
{
t=t->left;
}
else
{
t=t->right;
}
}
if(strcmp(x->name,e) > 0)
{
x->left = (tree*)malloc(sizeof(tree));
strcpy(x->left->name,e);
x->left->left=NULL;
x->left->right=NULL;
}
else
{
x->right = (tree*)malloc(sizeof(tree));
strcpy(x->right->name,e);
x->right->left=NULL;
x->right->right=NULL;
}
}
}
void inorder(tree *r)
{
if(r != NULL)
{
inorder(r->left);
printf("\t%s",r->name);
inorder(r->right);
}
```

```
}
int main()
{
insert("Anaka");
insert("Zen");
insert("Angel");
insert("Leya");
insert("Eldho");
printf("\nIn order: ");
inorder(root);
return 0;
}
```

Output

```
In order:      Anaka   Angel   Eldho   Leya    Zen
--------------------------------
```

Program#45
Representation of Graph using array

Source Code:

```
#include <stdio.h>
#include <stdlib.h>
struct AdjListNode
{
int dest;
struct AdjListNode* next;
};
struct AdjList
{
struct AdjListNode *head;
};
struct Graph
{
int V;
struct AdjList* array;
};
struct AdjListNode* newAdjListNode(int dest)
{
struct AdjListNode* newNode =
(struct AdjListNode*) malloc(sizeof(struct AdjListNode));
newNode->dest = dest;
newNode->next = NULL;
return newNode;
}
struct Graph* createGraph(int V)
{
struct Graph* graph =
(struct Graph*) malloc(sizeof(struct Graph));
graph->V = V;
graph->array=(struct AdjList*) malloc(V * sizeof(struct AdjList));
int i;
```

```
for (i = 0; i < V; ++i)
graph->array[i].head = NULL;
return graph;
}
void addEdge(struct Graph* graph, int src, int dest)
{
struct AdjListNode* newNode = newAdjListNode(dest);
newNode->next = graph->array[src].head;
graph->array[src].head = newNode;
newNode = newAdjListNode(src);
newNode->next = graph->array[dest].head;
graph->array[dest].head = newNode;
}
void printGraph(struct Graph* graph)
{
int v;
for (v = 0; v < graph->V; ++v)
{
struct AdjListNode* pCrawl = graph->array[v].head;
printf("\n Adjacency list of vertex %d\n head ", v);
while (pCrawl)
{
printf("-> %d", pCrawl->dest);
pCrawl = pCrawl->next;
}
printf("\n");
}
}
int main()
{
int V = 5;
struct Graph* graph = createGraph(V);
addEdge(graph, 0, 1);
```

```
addEdge(graph, 0, 4);
addEdge(graph, 1, 2);
addEdge(graph, 1, 3);
addEdge(graph, 1, 4);
addEdge(graph, 2, 3);
addEdge(graph, 3, 4);
printGraph(graph);
return 0;
}
```

Output

```
Adjacency list of vertex 0
head -> 4-> 1

Adjacency list of vertex 1
head -> 4-> 3-> 2-> 0

Adjacency list of vertex 2
head -> 3-> 1

Adjacency list of vertex 3
head -> 4-> 2-> 1

Adjacency list of vertex 4
head -> 3-> 1-> 0

--------------------------------
```

PROJECT
WORDLE- Word guessing game

INTRODUCTION

Wordle is a word guessing game using programming in C. Here LinkedList data structure is used to store and compare strings.

FUNCTIONALITY

First the quiz master is allowed to enter a 5 letter secret word. He will also be giving a hint related to the secret word. The player needs to guess the word within 5 attempts.

Source Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int flag=0;
int count;
int i,j;
char a[5], a1[5], c[5],str[20];
struct node
{
char data;
struct node *next;
};
struct node *list1 = NULL;
struct node *list2 = NULL;
void insert(struct node **list, char e)//LinkedList Insertion
{
struct node *t;
if (*list == NULL)
{
*list = (struct node *)malloc(sizeof(struct node));
(*list)->data = e;
(*list)->next = NULL;
}
else
```

```
{

t = *list;
while (t->next != NULL)
{
t = t->next;
}
t->next = (struct node *)malloc(sizeof(struct node));
t->next->data = e;
t->next->next = NULL;
}
}
int compare()// Comparing user input with secret word
{
struct node *t1 = list1;
struct node *t2 = list2;
int count=0;
while (t1 != NULL && t2 != NULL)
{
if (t1->data == t2->data)
{
printf("%c", t2->data);
t1 = t1->next;
t2 = t2->next;
count=count+1;
}
else
{
t2->data = 'x';

printf("%c", t2->data);
t2 = t2->next;
}
```

```
}
return(count);
}
void guess()
{
int i,j;
printf("\nGuess the word:");
scanf("%s", a1);
printf("Guessed word is: ");
for (j = 0; j < 5; j++)
{
insert(&list2, a1[j]);
}

}
void deleteLinkedList(struct node **list) //deleting the user string if it's wrong
{
struct node *current = *list;
struct node *next;
while (current != NULL)
{
next = current->next;
free(current);

current = next;
}
*list = NULL;
}
int main()
{
struct node *t2;
printf("Enter a 5-letter secret word: _ _ _ _ _\n");
scanf("%s", a);
```

```
//printf("\nData in linked list:");
for (i = 0; i < 5; i++)
{
insert(&list1, a[i]);
}
printf("\nEnter a hint:");
scanf(" %[^\n]s", str);

printf("\n\n***************Game Rules**************\nGuess the
secret word within 5 successful attempts!.");
printf("\n\n************* Game Starts ******************\n");
printf("Hint: %s\n", str);
for(i=0;i<5;i++)
{
guess();
count=compare();
if(count==5)
{

printf("\n\nCongratulation, correct guess");
break;
}
else
{
printf("\n\nWrong guess");
deleteLinkedList(&list2);
}
}
return 0;
}
```

OUTPUT:

E:\c\C project\wordle game mimic.exe

```
Enter a 5-letter secret word: _ _ _ _ _
clock

Enter a hint:My hands are just as important as my face, and I'm not one to sit still. What am I?


***************Game Rules**************
Guess the secret word within 5 successful attempts!.

************* Game Starts ********************
Hint: My hands are just as important as my face, and I'm not one to sit still. What am I?

Guess the word:watch
Guessed word is: xxxcx

Wrong guess
Guess the word:phone
Guessed word is: xxxxx

Wrong guess
Guess the word:close
Guessed word is: cloxx

Wrong guess
Guess the word:clock
Guessed word is: clock

Congratulation, correct guess
--------------------------------
Process exited after 62.73 seconds with return value 0
Press any key to continue . . .
```