DS LAB CYCLE

Section 1: Basics

1. Write programs to demonstrate the use of storage classes in C.

```
#include<stdio.h>

int a=3,e;//global variable

void stat()
{
        static int c,d=1;
        int i;
        printf("\nIn stat\n");
        for(i=0;i<5;i++)
        {
                printf("%d %d\n",c++,d++);
        }
}

void reg()
{
        int f=3,g=4;
        a=f+g;//register
        a=a+10;
        printf("Now value of a is %d",a);
}

int main()
{
        int b=3,h,k;
        printf("\nLocal Value of b:%d",b);
        printf("\nLocal Value of h:%d (not initialized)",h);//local variable is not initialized by
default it gives garbage value
        printf("\nLocal Value of k:%d (not initialized)",k);
        printf("\nGlobal value of a:%d",a);
        printf("\nGlobal value of e:%d (not initialized)",e);//global variable by default is 0

        stat();
        reg();
}
```

2. Use a menu-driven program to insert, search, delete and sort elements in an array using functions (use global variables)

```
#include<stdio.h>

int arr[10];
int n,i,j,ch,k,flag=0,temp;

void insert()
{
	printf("Enter the elements: ");
	for(i=0;i<n;i++)
	{
		scanf("%d",&arr[i]);
	}
}

void search()
{
	printf("Enter the element to be searched: ");
	scanf("%d",&k);
	for(i=0;i<n;i++)
	{
		if(k==arr[i])
		{
			printf("Element found at position %d",i);
			flag=1;
			break;
		}

	}
	if(flag!=1)
	{
		printf("\nElement not found");
	}
}

void sort()
{
	for(i=0;i<n;i++)
	{
		for(j=0;j<n;j++)
		{
			if(arr[i]>arr[j])
```

```
                        {
                                temp=arr[i];
                                arr[i]=arr[j];
                                arr[j]=arr[i];
                                printf("%d",arr[i]);
                        }
                }
        }
}
void display()
{
        printf("Array Elements are: ");
        for(i=0;i<n;i++)
        {
                printf("%d ",arr[i]);
        }
}

int menu()
{
        printf("\n1.Insert\n2.Search\n3.Delete\n4.Sort\n5.Display\n6.Exit\nEnter your
choice: ");
        scanf("%d",&ch);
        return ch;
}

void process()
{
        for(ch=menu();ch!=6;ch=menu())
        {
                switch(ch)
                {
                        case 1:insert();
                                break;
                        case 2:search();
                                break;
                        case 3:printf("--");
                                break;
                        case 4:sort();
                                break;
                        case 5:display();
                                break;
                        default: printf("\nInvalid Choice");
                }
```

```
        }
}

int main()
{
        printf("Enter the size of array: ");
        scanf("%d",&n);
        process();
        return 0;
}
```

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

```
#include<stdio.h>


void insert(int arr[],int n)
{
   int i;
        printf("Enter the elements: ");
        for(i=0;i<n;i++)
        {
                scanf("%d",&arr[i]);
        }
}

void search(int arr[],int n)
{
   int i,k,flag=0;
        printf("Enter the element to be searched: ");
        scanf("%d",&k);
        for(i=0;i<n;i++)
        {
                if(k==arr[i])
                {
                        printf("Element found at position %d",i);
                        flag=1;
                        break;
                }

        }
        if(flag!=1)
```

```
        {
                printf("\nElement not found");
        }
}

void delete(int arr[],int n)
{
   int i,j,del;
   printf("Enter the element to be deleted: ");
   scanf("%d",&del);
   for (i = 0; i < n; i++)
   {
      if (del == arr[i])
      {
         for (j = i; j < n - 1; j++)
         {
            arr[j] = arr[j + 1];
         }
         n--;
         printf("Element %d deleted successfully.", del);
         return;
      }
   }
   printf("Element %d not found in the array.", del);
}

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

                        }
                }
        }
        printf("Sorted Array Elements are: ");
        for(i=0;i<n;i++)
```

```
        {
                printf("%d ",arr[i]);
        }
}

void display(int arr[],int n)
{
   int i;
        printf("Array Elements are: ");
        for(i=0;i<n;i++)
        {
                printf("%d ",arr[i]);
        }
}

int menu()
{
   int ch;
        printf("\n1.Insert\n2.Search\n3.Delete\n4.Sort\n5.Display\n6.Exit\nEnter your
choice: ");
        scanf("%d",&ch);
        return ch;
}

void process(int arr[],int n)
{
   int ch;
        for(ch=menu();ch!=6;ch=menu())
        {
                switch(ch)
                {
                        case 1:insert(arr,n);
                                break;
                        case 2:search(arr,n);
                                break;
                        case 3:delete(arr,n);
                                break;
                        case 4:sort(arr,n);
                                break;
                        case 5:display(arr,n);
                                break;
                        default: printf("\nInvalid Choice");
                }
        }
```

```
}

int main()
{
   int arr[10],n;
         printf("Enter the size of array: ");
         scanf("%d",&n);
         process(arr,n);
         return 0;
}
```

4. Search for all the occurrences of an element in an integer array (positions)

```
#include<stdio.h>

int arr[10],i,ele;

void read(int n)
{
   printf("\nEnter the array elements : ");
   for(i=0;i<n;i++)
   {
      scanf("%d",&arr[i]);
   }
}

void disp(int n)
{
   printf("\nThe array elements are: ");
   for(i=0;i<n;i++)
   {
      printf("%d\t",arr[i]);
   }
}

void search(int n)
{
   int ele,m=0;
   printf("\nEnter the element to be searched for: ");
   scanf("%d",&ele);
   for(i=0;i<n;i++)
   {
      if(arr[i]==ele)
```

```
        {
            printf("\nElement found at position %d",i);
            m=m+1;
        }
    }
    if(m>0)
    {
        printf("\nElement %d has %d occurence",ele,m);
    }
    else
    {
        printf("\nElement not found");
    }
}

int main()
{
    int n;
    printf("Enter the limit of the array: ");
    scanf("%d",&n);
    read(n);
    disp(n);
    search(n);
}
```

5. Sort the array elements in ascending order (minimum three functions: read, disp and sort)

```
#include<stdio.h>

int arr[10],i,j,temp;

void read(int n)
{
    printf("\nEnter the array elements : ");
    for(i=0;i<n;i++)
    {
        scanf("%d",&arr[i]);
    }
}

void disp(int n)
{
```

```
    printf("\nThe array elements are: ");
    for(i=0;i<n;i++)
    {
        printf("%d\t",arr[i]);
    }
}

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

                          }
                  }
          }
          printf("\nSorted Array Elements are: ");
          for(i=0;i<n;i++)
          {
                  printf("%d\t",arr[i]);
          }
}


int main()
{
    int n;
    printf("Enter the limit of the array: ");
    scanf("%d",&n);
    read(n);
    disp(n);
    sort(n);
}
```

6. Two-dimensional matrix: using functions

a. Addition

b. Subtraction

c. Multiplication

d. Transpose [Need to discuss before implementing]

e. Determinant

```
#include<stdio.h>

int arr1[10][10],arr2[10][10],arr3[10][10],i,j,k;

void read(int r1,int c1,int r2,int c2)
{
   printf("\nEnter the elements of 1st matrix: ");
   for(i=0;i<r1;i++)
   {
      for(j=0;j<c1;j++)
      {
         scanf("%d",&arr1[i][j]);
      }
   }
   printf("\nEnter the elements of 2nd matrix: ");
   for(i=0;i<r2;i++)
   {
      for(j=0;j<c2;j++)
      {
         scanf("%d",&arr2[i][j]);
      }
   }
}

void disp(int r1,int c1,int r2,int c2)
{
         printf("\nThe elements of 1st matrix are: \n");
         for(i=0;i<r1;i++)
         {
                  for(j=0;j<c1;j++)
                  {
                  printf("%d ",arr1[i][j]);
         }
```

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

                printf("\nThe elements of 2nd matrix are: \n");
        for(i=0;i<r2;i++)
        {
                for(j=0;j<c2;j++)
        {
          printf("%d ",arr2[i][j]);
        }
        printf("\n");
        }
}

void add(int r1,int c1,int r2,int c2)
{
   printf("\nAddition: \n");
   if(r1==r2 && c1==c2)
   {
        {
                for(i=0;i<r1;i++)
                {
                for(j=0;j<c1;j++)
                {
                        arr3[i][j]=arr1[i][j]+arr2[i][j];
                        printf("%d\t",arr3[i][j]);
                }
        printf("\n");
                }
        }
        }
        else
        {
                printf("For addition matrix should be of same size");
        }
}


void sub(int r1,int c1,int r2,int c2)
{

   printf("\nSubtraction: \n");
   if(r1==r2 && c1==c2)
   {
```

```
        for(i=0;i<r1;i++)
        {
        for(j=0;j<c1;j++)
        {
                arr3[i][j]=arr1[i][j]-arr2[i][j];
                printf("%d\t",arr3[i][j]);
        }
        printf("\n");
        }
        }
        else
        {
                printf("For subtraction matrix should be of same size");
        }
}

void mul(int r1,int c1,int r2,int c2)
{
   printf("\nMultiplication: \n");
   if(c1==r2)
   {
        for(i=0;i<r1;i++)
        {
        for(j=0;j<c2;j++)
        {
                arr3[i][j]=0;
                for(k=0;k<c2;k++)
                {
                        arr3[i][j]=arr3[i][j]+ arr1[i][k] * arr2[k][j];

                }
                printf("%d\t",arr3[i][j]);
        }
      printf("\n");
        }
        }
        else
        {
                printf("Multiplication not possible the size should be of the
form((axb)(bxc))");
        }
}

void tran(int r1,int c1,int r2,int c2)
```

```
{
   printf("\nTransponse of 1st matrix: \n");
   for(i=0;i<c1;i++)
   {
      for(j=0;j<r1;j++)
      {
         printf("%d ",arr1[j][i]);
      }
      printf("\n");
   }
   printf("\nTransponse of 2nd matrix: \n");
   for(i=0;i<c2;i++)
   {
      for(j=0;j<r2;j++)
      {
         printf("%d ",arr2[j][i]);
      }
      printf("\n");
   }
}

int menu()
{
         int ch;

printf("\nChoices:\n1.Display\n2.Addition\n3.Subtraction\n4.Multiplication\n5.Transpose\n
6.Exit\nEnter your choice: ");
         scanf("%d",&ch);
         return ch;
}

void process(int r1,int c1,int r2,int c2)
{
         int ch;
         for(ch=menu();ch!=6;ch=menu())
         {
                  switch(ch)
                  {

                           case 1: disp(r1,c1,r2,c2);
                                    break;
                           case 2: add(r1,c1,r2,c2);
                                    break;
                           case 3: sub(r1,c1,r2,c2);
```

```
                                        break;
                                case 4: mul(r1,c1,r2,c2);
                                        break;
                                case 5: tran(r1,c1,r2,c2);
                                        break;
                                default:printf("Invalid Choice");
                                break;

                        }
                }
}
int main()
{
   int r1,c1,r2,c2;
   printf("Enter the row size and column size of the 1st array: ");
   scanf("%d %d",&r1,&c1);
   printf("Enter the row size and column size of the 2nd array: ");
   scanf("%d %d",&r2,&c2);
   read(r1,c1,r2,c2);
   process(r1,c1,r2,c2);
   /*
         disp(r,c);
         tran(r,c);
         add(r,c);
         sub(r,c);
         mul(r,c);
   */

}
```

7. Display the array elements in the same order using a recursive function

```
#include <stdio.h>

void read(int arr[],int n)
{
   int i;
   printf("Enter the array elements: ");
   for(i=0;i<n;i++)
   {
      scanf("%d",&arr[i]);
   }
}
```

```
void dispRec(int arr[],int n,int index)
{
   if(index==n)
   {
      return;
   }
   else
   {
      printf("%d ", arr[index]);
      dispRec(arr, n, index + 1);
   }
}

int main()
{
   int n,arr[10],index=0;
   printf("Enter the limit of the array: ");
   scanf("%d",&n);

   read(arr,n);

   printf("\nArray Elements are: ");
   dispRec(arr,n,index);
}
```

8. Display array elements in reverse order using a recursive function

```
#include <stdio.h>

void read(int arr[],int n)
{
   int i;
   printf("Enter the array elements: ");
   for(i=0;i<n;i++)
   {
      scanf("%d",&arr[i]);
   }
}

void dispRec(int arr[],int n,int index)
{
   if(index < 0)
```

```
    {
        return;
    }
    else
    {
        printf("%d ", arr[index]);
        dispRec(arr, n, index-1);
    }
}

int main()
{
    int n,arr[10];
    printf("Enter the limit of the array: ");
    scanf("%d",&n);

    read(arr,n);

    printf("\nArray Elements are: ");
    dispRec(arr,n,n-1);
}
```

Section 2: Stack

9. Implement stack operations using arrays.

```
#include<stdio.h>
int stack[5],top=-1;

void push(int e)
{
        if( top+1==5)
        {
                printf("Error:Stack is Full");
        }
        else
        {
                stack[++top]=e;
        }
}

void pop()
```

```
{
        if(top==-1)
        {
                printf("Error: Stack is empty");
        }
        else
        {
                printf("Deleted Element: %d",stack[top--]);
        }
}

void disp()
{
   int i;
   if (top == -1)
   {
      printf("Stack is Empty");
   }
   else
   {
      printf("Stack elements are:  ");
      for ( i = top; i >= 0; i--)
      {
         printf("%d ", stack[i]);
      }
   }
   printf("\n");
}

int menu()
{
        int ch;
        printf("\n1.Push\n2.Pop\n3.Display\n4.Exit\nEnter your choice: ");
        scanf("%d",&ch);
        return ch;
}

void process()
{
        int ch;
        for(ch=menu();ch!=4;ch=menu())
        {
                switch(ch)
                {
```

```
                        case 1: printf("\nEnter the element: ");
                                scanf("%d",&ch);
                                push(ch);
                                break;

                        case 2: pop();
                                break;

                        case 3: disp();
                               break;

                        default: printf("Invalid Choice");
                }
        }
}

int main()
{
        process();
        return 0;
}
```

10. Reverse a string using Stack

```
#include<stdio.h>

#include<string.h>

int stack[20],top=-1;

char a[20];

int i;

void pop()

{

   printf("\nReversed String : ");

   for(i=top;top>=-1;top--)

   {
```

```
            printf("%c",stack[top]);
        }
    }
    void push()

    {

        for(i=0;i<strlen(a);i++)
        {
            top++;
            stack[top]=a[i];
        }
        pop();
    }

    void main()
    {
        printf("\nEnter the string : ");
        gets(a);
        push(a);
    }
```

Session 3: Stack

11. Convert an expression from infix to postfix using stack

12. Evaluate an expression using stack

Session 4: Struct

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

```
#include<stdio.h>

struct date
{
   int date1,month1,year1;
   int date2,month2,year2;
};

struct date d;

void read()
{
   printf("Enter the 1st Date(dd-mm-yyyy): ");
   scanf("%d-%d-%d",&d.date1,&d.month1,&d.year1);
   printf("Enter the 2nd Date(dd-mm-yyyy): ");
   scanf("%d-%d-%d",&d.date2,&d.month2,&d.year2);
}

void disp()
{
   printf("\n1st Date: %d-%d-%d",d.date1,d.month1,d.year1);
   printf("\n2nd Date: %d-%d-%d",d.date2,d.month2,d.year2);
}

void compare()
{
   if(d.date1==d.date2&&d.month1==d.month2&&d.year1==d.year2)
   {
      printf("\nTwo Dates are Same");
   }
   else
   {
      printf("\nTwo Dates are not Same");
   }
```

```
}

int main()
{
   read();
   disp();
   compare();
}
```

14. Define a structure for employees with eno,ename, esal and dno. Read n employees information and provide functions for the following:

a. Searching an employee by no

b. Sorting the employees by

i. Name

ii. Salary

c. Deleting an employee

```
#include <stdio.h>
#include <string.h>  // Added to use strcmp function

struct emp
{
   int eno, dno;
   char ename[20];
   float esal;
};

struct emp e[10];

int i;

void read(int n)
{
   printf("\nEnter the Employee Details\n");
   for ( i = 0; i < n; i++)
   {
      printf("Employee : %d\n", i + 1);  // Added a newline character
      printf("Enter the Employee No: ");
      scanf("%d", &e[i].eno);
      printf("Enter the Employee Name: ");
      scanf("%s", e[i].ename);
      printf("Enter the Employee Salary: ");
      scanf("%f", &e[i].esal);
```

```
        printf("Enter the DNo: ");
        scanf("%d", &e[i].dno);
    }
}

void disp(int n)
{
    printf("\nEmployee Details are: \n");
    for ( i = 0; i < n; i++)
    {
         printf("\n%d:Employee No: %d\nEmployee Name: %s\nEmployee Salary: %f\nDNo:
%d\n", i + 1, e[i].eno, e[i].ename, e[i].esal, e[i].dno);
    }
}

void search(int n)
{
    int srch, flag = 0;
    printf("\nEnter the employee no. to be searched: ");
    scanf("%d", &srch);
    for ( i = 0; i < n; i++)
    {
        if (srch == e[i].eno)
        {
            flag = 1;
              printf("\nEmployee No: %d\nEmployee Name: %s\nEmployee Salary: %f\nDNo:
%d\n", e[i].eno, e[i].ename, e[i].esal, e[i].dno);
            break;
        }
    }
    if (flag == 0)
    {
        printf("\nEmployee not found");
    }
}

void sort(int n)
{
    int ch, j;
    struct emp temp;

    printf("Sort on the basis of \n(1)Name\n(2)Salary\nEnter your choice: ");
    scanf("%d", &ch);
    switch (ch)
```

```
   {
   case 1:
      for ( i = 0; i < n; i++)
      {
         for (j = i + 1; j < n; j++)
         {
            if (strcmp(e[i].ename, e[j].ename) > 0)
            {
               temp = e[i];
               e[i] = e[j];
               e[j] = temp;
            }
         }
      }
        printf("Employees sorted on the basis of name: ");  // Added a message for sorting
by name
      for ( i = 0; i < n; i++)
      {
            printf("\nEmployee No: %d\nEmployee Name: %s\nEmployee Salary: %f\nDNo:
%d\n", e[i].eno, e[i].ename, e[i].esal, e[i].dno);
      }
      break;

   case 2:
      for ( i = 0; i < n; i++)
      {
         for (j = i + 1; j < n; j++)
         {
            if (e[i].esal > e[j].esal)
            {
               temp = e[i];
               e[i] = e[j];
               e[j] = temp;
            }
         }
      }
      printf("Employees sorted on the basis of salary: ");
      for ( i = 0; i < n; i++)
      {
            printf("\nEmployee No: %d\nEmployee Name: %s\nEmployee Salary: %f\nDNo:
%d\n", e[i].eno, e[i].ename, e[i].esal, e[i].dno);
      }
      break;
   default:
```

```
        printf("\nInvalid Choice\n");
        break;
    }
}

void delete(int *n)
{
    int emp, i, found = 0, index, temp = 0;
    printf("\nEnter the employee no to be deleted: ");
    scanf("%d", &emp);
    for (i = 0; i < *n; i++)
    {
        if (emp == e[i].eno)
        {
            found = 1;
            index = i;
            break;
        }
    }
    if (found == 1)
    {
        for (i = index; i < *n - 1; i++)
        {
            e[i] = e[i + 1];
        }
        (*n)--;
        printf("\nEmployee with Employee No: %d has been deleted\n", emp);
        disp(*n);
    }
    else
    {
        printf("\nEmployee with Employee No: %d not found\n", emp);
    }
}

void process(int n)
{
    int c;
    for (c = menu(); c != 5; c = menu())
    {
        switch (c)
        {
        case 1:
            disp(n);
```

```
            break;
        case 2:
            search(n);
            break;
        case 3:
            sort(n);
            break;
        case 4:
            delete(&n);
            break;
        default:
            printf("\nInvalid Choice");
            break;
        }
    }
}

int menu()
{
    int c;
        printf("\nChoices are: \n1.Display\n2.Search\n3.Sort\n4.Delete\n5.Exit\nEnter your
choice: ");
    scanf("%d", &c);
    return c;
}

int main()
{
    int n;
    printf("Enter the no.of Employees: ");
    scanf("%d", &n);
    read(n);
    process(n);
}
```

Session 5: Polynomials using Array

15. Read a polynomial and display it; use array

```
#include<stdio.h>
int coeff1[10];
int i;
int read(int degree)
```

```
{
        printf("Enter the elements the polynomial: \n");
        for(i=degree;i>=0;i--)
        {
                if(i==0)
                {
                        printf("Enter the constant term:",i);
                        scanf("%d",&coeff1[i]);
                        break;
                }
                printf("Enter the coefficient of x^%d:",i);
                scanf("%d",&coeff1[i]);
        }
}

int disp(int degree)
{
   printf("The polynomial is: \n");
   for(i=degree;i>=0;i--)
        {
                if(i==0)
                {
                        printf("%d",coeff1[i]);
                        break;
                }
                printf("%dx^%d+",coeff1[i],i);
        }
}
int main()
{
        int degree;
        printf("Enter the degree of the polynomial: ");
        scanf("%d",&degree);
        read(degree);
        disp(degree);
}
```

16. Add two polynomials using the array itself

```
#include<stdio.h>

int coeff1[10];
        int coeff2[10];
```

```
        int coeff3[10];
        int i;

int read(degree)
{
        printf("Enter the elements of 1st polynomial: \n");
        for(i=degree;i>=0;i--)
        {
                if(i==0)
                {
                        printf("Enter the constant term:",i);
                        scanf("%d",&coeff1[i]);
                        break;
                }
                printf("Enter the coefficient of x^%d:",i);
                scanf("%d",&coeff1[i]);
        }
        printf("Enter the elements of 2nd polynomial: \n");
        for(i=degree;i>=0;i--)
        {
                if(i==0)
                {
                        printf("Enter the constant term:",i);
                        scanf("%d",&coeff2[i]);
                        break;
                }
                printf("Enter the coefficient of x^%d:",i);
                scanf("%d",&coeff2[i]);
        }
}

int disp(degree)
{
   printf("1st polynomial: \t");
        for(i=degree;i>=0;i--)
        {
                if(i==0)
                {
                        printf("%d",coeff1[i]);
                        break;
                }
                printf("%dx^%d+",coeff1[i],i);
        }
        printf("\n2nd polynomial: \t");
```

```
        for(i=degree;i>=0;i--)
        {
                if(i==0)
                {
                        printf("%d",coeff2[i]);
                        break;
                }
                printf("%dx^%d+",coeff2[i],i);
        }
}

int add(degree)
{
   printf("\nAddition of 1st and 2nd polynomial\t");
        for(i=degree;i>=0;i--)
        {
                coeff3[i]=coeff1[i]+coeff2[i];
        }
        for(i=degree;i>=0;i--)
        {
                if(i==0)
                {
                        printf("%d",coeff3[i]);
                        break;
                }
                printf("%dx^%d+",coeff3[i],i);
        }
}
int main()
{
        int degree;
        printf("Enter the degree of the polynomial: ");
        scanf("%d",&degree);
        read(degree);
        disp(degree);
        add(degree);
}
```

Session 6: Polynomials using Structure

17. Read a polynomial and display it; use structure array

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

```
int i;

struct poly
{
   int coeff;
};

struct poly p[10];

int read(int degree)
{

        printf("Enter the elements of polynomial: \n");
        for(i=degree;i>=0;i--)
        {
                if(i==0)
                {
                        printf("Enter the constant term:",i);
                        scanf("%d",&p[i].coeff);
                }
                else
                {
                   printf("Enter the coefficient of x^%d:",i);
                   scanf("%d",&p[i].coeff);
                }
        }
}

int disp(int degree)
{

   printf("The Polynomial is: \n");
        for(i=degree;i>=0;i--)
        {
                if(i==0)
                {
                        printf("%d",p[i].coeff);
                }
                else
                {
                   printf("%dx^%d+",p[i].coeff,i);
           }
        }
```

```
}



int main()
{
        int degree;
        printf("Enter the degree of the polynomial: ");
        scanf("%d",&degree);
        read(degree);
        disp(degree);
}
```

18. Add two polynomials

```
#include<stdio.h>

int i;

struct poly
{
   int coeff1,coeff2,coeff3;
};

struct poly p[10];
void read(int degree)
{
        printf("Enter the elements of 1st polynomial: \n");
        for(i=degree;i>=0;i--)
        {
                if(i==0)
                {
                        printf("Enter the constant term:",i);
                        scanf("%d",&p[i].coeff1);
                }
                else
                {
                   printf("Enter the coefficient of x^%d:",i);
                   scanf("%d",&p[i].coeff1);
                }
        }
        printf("Enter the elements of 2nd polynomial: \n");
        for(i=degree;i>=0;i--)
```

```
        {
                if(i==0)
                {
                        printf("Enter the constant term:",i);
                        scanf("%d",&p[i].coeff2);
                }
                else
                {
                   printf("Enter the coefficient of x^%d:",i);
                   scanf("%d",&p[i].coeff2);
                }
        }
}

void disp(int degree)
{
   printf("The 1st Polynomial is: \t");
        for(i=degree;i>=0;i--)
        {
                if(i==0)
                {
                        printf("%d",p[i].coeff1);
                }
                else
                {
                   printf("%dx^%d+",p[i].coeff1,i);
           }
        }
        printf("\nThe Polynomial is:\t");
        for(i=degree;i>=0;i--)
        {
                if(i==0)
                {
                        printf("%d",p[i].coeff2);
                }
                else
                {
                   printf("%dx^%d+",p[i].coeff2,i);
           }
        }
}

void add(degree)
{
```

```
    printf("\nAddition of 1st and 2nd polynomial:\t");
        for(i=degree;i>=0;i--)
        {
                p[i].coeff3=p[i].coeff1+p[i].coeff2;
        }
        for(i=degree;i>=0;i--)
        {
                if(i==0)
                {
                        printf("%d",p[i].coeff3);
                        break;
                }
                printf("%dx^%d+",p[i].coeff3,i);
        }
}


int main()
{
        int degree;
        printf("Enter the degree of the polynomial: ");
        scanf("%d",&degree);
        read(degree);
        disp(degree);
        add(degree);

}
```

19. Subtract two polynomials

```
#include<stdio.h>

int i;

struct poly
{
   int coeff1,coeff2,coeff3;
};

struct poly p[10];
void read(int degree)
{
        printf("Enter the elements of 1st polynomial: \n");
```

```
        for(i=degree;i>=0;i--)
        {
                if(i==0)
                {
                        printf("Enter the constant term:",i);
                        scanf("%d",&p[i].coeff1);
                }
                else
                {
                   printf("Enter the coefficient of x^%d:",i);
                   scanf("%d",&p[i].coeff1);
                }
        }
        printf("Enter the elements of 2nd polynomial: \n");
        for(i=degree;i>=0;i--)
        {
                if(i==0)
                {
                        printf("Enter the constant term:",i);
                        scanf("%d",&p[i].coeff2);
                }
                else
                {
                   printf("Enter the coefficient of x^%d:",i);
                   scanf("%d",&p[i].coeff2);
                }
        }
}

void disp(int degree)
{
   printf("The 1st Polynomial is: \t");
        for(i=degree;i>=0;i--)
        {
                if(i==0)
                {
                        printf("%d",p[i].coeff1);
                }
                else
                {
                   printf("%dx^%d+",p[i].coeff1,i);
              }
        }
        printf("\nThe Polynomial is:\t");
```

```
        for(i=degree;i>=0;i--)
        {
                if(i==0)
                {
                        printf("%d",p[i].coeff2);
                }
                else
                {
                   printf("%dx^%d+",p[i].coeff2,i);
              }
        }
}


void sub(degree)
{
   printf("\nSubtraction of 1st and 2nd polynomial:\t");
        for(i=degree;i>=0;i--)
        {
                p[i].coeff3=p[i].coeff1-p[i].coeff2;
        }
        for(i=degree;i>=0;i--)
        {
                if(i==0)
                {
                        printf("%d",p[i].coeff3);
                        break;
                }
                printf("%dx^%d+",p[i].coeff3,i);
        }
}


int main()
{
        int degree;
        printf("Enter the degree of the polynomial: ");
        scanf("%d",&degree);
        read(degree);
        disp(degree);
        sub(degree);

}
```

20. Multiply two polynomials

Session 7: Dynamic Memory Allocation

21. Implement

a) malloc b) calloc and c) free functions

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

int main() {
   int *p, *q;

   // Allocate memory using malloc
   p = (int *)malloc(sizeof(int));

   // Allocate memory using calloc
   q = (int *)calloc(1, sizeof(int));


   printf("In malloc Before Initialization Value of *p is undefined\n");
   printf("Enter a number: ");
   scanf("%d", p);
   printf("After Initialization Value of *p is %d\n", *p);

   printf("\nIn calloc Before Initialization Value of *q is %d\n", *q);
   printf("Enter a number: ");
   scanf("%d", q);
   printf("After Initialization Value of *q is %d\n", *q);

   free(p);
   free(q);
   return 0;
}
```

22. Use malloc to read n integers and find the mean.

23. Use calloc to read n numbers and find the mode.

24. 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.

25. Use realloc to implement varchar for any length.

Session 8- Queue

26. Implement Queue using array

```
#include <stdio.h>

int f = -1, r = -1;
int q[5];

void enqueue(int e) // inserting an element into the queue
{
   if (((r + 1) % 5) == f) // Check if the queue is full
   {
      printf("\nQueue is full");
   }
   else
   {
      if (f == -1)
      {
         f = 0;
      }
      r = (r + 1) % 5;
      q[r] = 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;
      }
   }
```

```
}

void displayQueue()
{
   if (f == -1)
   {
      printf("\nQueue is empty");
   }
   else
   {
      int i = f;
      printf("\nQueue elements: ");
      while (1)
      {
         printf("%d ", q[i]);

         if (i == r)
         {
            break;
         }

         i = (i + 1) % 5;
      }
      printf("\n");
   }
}

void process(int c)
{
   int e;
   for (c = menu(); c != 4; c = menu())
   {
      switch (c)
      {
      case 1: printf("Enter the element to be inserted: \n");
            scanf("%d",&e);
            enqueue(e);
            break;
      case 2: dequeue();
            break;
      case 3: displayQueue();
            break;
      default:printf("\nError: Invalid Choice");
            break;
```

```
        }
    }
}


int menu()
{
    int c;
    printf("\nChoices are: \n1.Enqueue\n2.Dequeue\n3.Display\n4.Exit\nEnter your choice:
");
    scanf("%d", &c);
    return c;
}

int main()
{
    int c;

    process(c);
}
```

27. Implement priority queue

Section 9- Linked List- Basics

1. Demonstrate a linked list creation and display
2. Write a program with functions to insert a new node
    a. at the beginning of a Singly Linked List.
    b. At the end of the linked list
    c. after a specified element in a linked list.
3. Write a program with functions to delete a node
    a. From the beginning of the linked list
    b. From the end of the linked list
    c. The node with specified data element
4. Write a program to create a singly linked list of n nodes and display it in reverse order.
5. Sort the elements in a linked list using

a. changing the values (swapping the values)

b. Changing the address (Swapping the address)

Section 10: Polynomial using Linked List

1. Polynomial using linked list - addition and multiplication
2. Linked list using names - insert, delete, display, sort, reverse, count

Section 11- linked list

1. Perform the respective operations on the following [Separate Question]

a. Linked Stack

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

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

typedef struct node stack;
enum{PUSH=1,POP,EXIT};
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("Empty Stack");
   }
   else
   {
      printf("\n%d",top->data);
      top=top->next;
   }
}

int menu()
{
   int ch;
   printf("\n1.PUSH\n2.POP\n3.Exit\nYour Choice: ");
   scanf("%d",&ch);
   return ch;
}

void process(int ch)
{
```

```
    int e;
    for(ch=menu();ch!=3;ch=menu())
    {
        switch(ch)
        {
            case PUSH:  printf("Enter the element to push: ");
                        scanf("%d",&e);
                        push(e);
                        break;
            case POP:   pop();
                        break;
            default:    printf("Error: Invalid Choice");
                        break;
        }
    }
}

int main()
{
    int ch;
    process(ch);
    return 0;
}
```

b. Linked Queue

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

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

typedef struct node queue;
enum{ENQUEUE=1,DEQUEUE,EXIT};
queue *f=NULL,*r=NULL;

void enqueue(int e)
{
    queue *t=(queue*)malloc(sizeof(queue));
    t->data=e;
```

```
    t->next=NULL;
    if(f==NULL)
    {
        f=t;
        r=t;
    }
    else
    {
        r->next=t;
        r=t;
    }
}

void dequeue()
{
    if(f==NULL)
    {
        printf("\nEmpty Queue");
    }
    else
    {
        printf("\n%d",f->data);
        f=f->next;
        if(f==NULL)
        {
            r=NULL;
        }
    }
}

int menu()
{
    int ch;
    printf("\n1.Enqueue\n2.Dequeue\n3.Exit\nEnter your choice: ");
    scanf("%d",&ch);
    return ch;
}

void process()
{
    int ch;
    for(ch=menu();ch!=3;ch=menu())
    {
        switch(ch)
```

```
        {
            case 1: printf("Enter the element to enqueue: ");
                    scanf("%d",&ch);
                    enqueue(ch);
                    break;

            case 2: dequeue();
                    break;

            default:printf("Error: Invalid Choice");
                    break;

        }
    }
}

int main()
{
    process();
    return 0;
}
```

c. Circular Linked List

d. Circular Linked Queue

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

// Structure to represent a node in the circular queue
struct Node {
    int data;
    struct Node* next;
};

// Initialize front and rear pointers to NULL
struct Node* front = NULL;
struct Node* rear = NULL;

// Function to check if the circular queue is empty
int isEmpty() {
    return front == NULL && rear == NULL;
}
```

```
// Function to enqueue an element into the circular queue
void enqueue(int value) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    if (!newNode) {
        printf("Memory allocation failed\n");
        exit(1);
    }
    newNode->data = value;
    newNode->next = NULL;

    if (isEmpty()) {
        front = rear = newNode;
        rear->next = front; // Make the circular connection
    } else {
        rear->next = newNode;
        rear = newNode;
        rear->next = front; // Make the circular connection
    }
    printf("Enqueued: %d\n", value);
}

// Function to dequeue an element from the circular queue
void dequeue() {
    if (isEmpty()) {
        printf("Queue is empty, cannot dequeue\n");
        return;
    }

    int removedValue = front->data;
    struct Node* temp = front;

    if (front == rear) {
        // If there is only one element in the queue
        front = rear = NULL;
    } else {
        front = front->next;
        rear->next = front; // Update the circular connection
    }

    free(temp);
    printf("Dequeued: %d\n", removedValue);
}
```

```
// Function to display the elements of the circular queue
void display() {
    if (isEmpty()) {
        printf("Queue is empty\n");
        return;
    }

    struct Node* current = front;
    printf("Circular Queue: ");
    do {
        printf("%d ", current->data);
        current = current->next;
    } while (current != front);
    printf("\n");
}

int main() {
    int choice, value;

    while (1) {
        printf("\nCircular Queue Operations:\n");
        printf("1. Enqueue\n");
        printf("2. Dequeue\n");
        printf("3. Display\n");
        printf("4. Quit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                printf("Enter value to enqueue: ");
                scanf("%d", &value);
                enqueue(value);
                break;
            case 2:
                dequeue();
                break;
            case 3:
                display();
                break;
            case 4:
                printf("Exiting program\n");
                exit(0);
            default:
```

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

        return 0;
    }
```

e. Doubly Linked List
f. Circular doubly linked list - store string values as data part

Section 12: Binary Search Tree

1. Binary search tree insertion and display Traversal using inorder, preorder and postorder using recursion
2. Binary search tree insertion and display in-order without using recursion
3. .Binary search tree insertion and display pre-order without using recursion
4. Binary search tree insertion and display post-order without using recursion
5. Binary search tree insertion using names and display the names in ascending order using inorder traversal.

Section -13: Graphs

1. Demonstrate the data structure of adjacent matrix using arrays
2. Demonstrate the data structure of adjacent matrix using linked lists