

DATA STRUCTURE PROGRAMMING

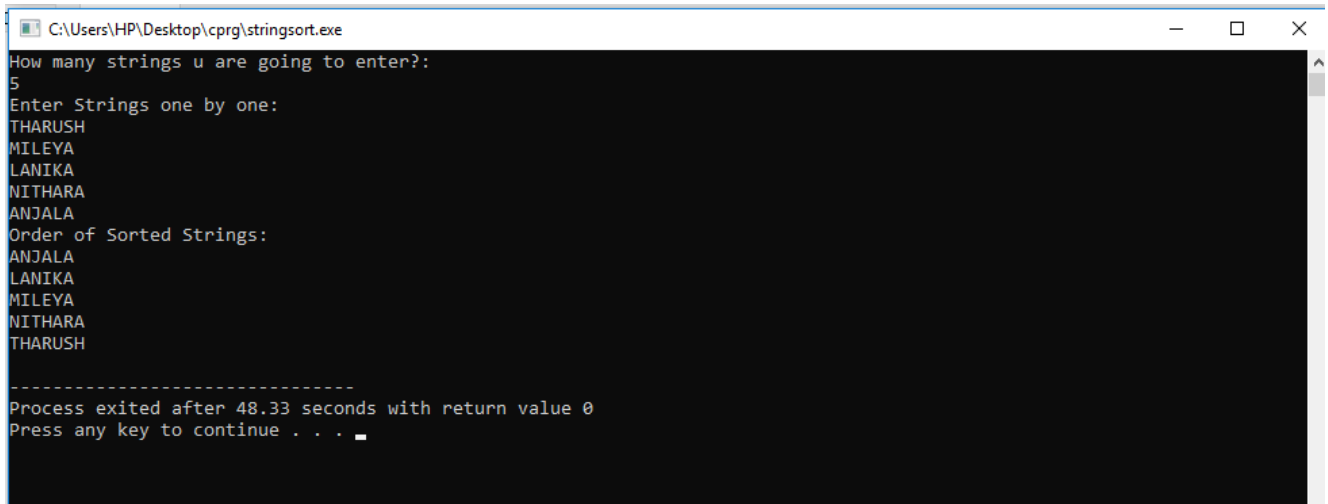
COMPUTER APPLICATION - IV SEMESTER

1. SORT A GIVEN LIST OF STRINGS

```
#include<stdio.h>
#include<string.h>
int main()
{
    int i,j,count;
    char str[25][25],temp[25];
    puts("How many strings u are going to enter?: ");
    scanf("%d",&count);
    puts("Enter Strings one by one: ");
    for(i=0;i<=count;i++)
        gets(str[i]);
    for(i=0;i<=count;i++)
        for(j=i+1;j<=count;j++)
        {
            if(strcmp(str[i],str[j])>0)
            {
                strcpy(temp,str[i]);
                strcpy(str[i],str[j]);
                strcpy(str[j],temp);
            }
        }
    printf("Order of Sorted Strings:");
    for(i=0;i<=count;i++)
        puts(str[i]);

    return 0;
}
```

OUTPUT



```
C:\Users\HP\Desktop\cprg\stringsort.exe
How many strings u are going to enter?:
5
Enter Strings one by one:
THARUSH
MILEYA
LANIKA
NITHARA
ANJALA
Order of Sorted Strings:
ANJALA
LANIKA
MILEYA
NITHARA
THARUSH
-----
Process exited after 48.33 seconds with return value 0
Press any key to continue . . .
```

2. REVERSE A STRING USING POINTER

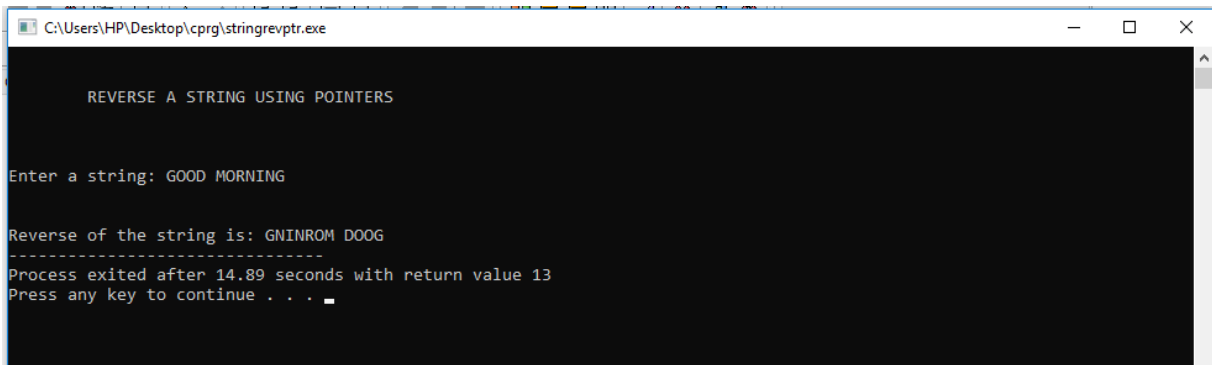
```
#include <stdio.h>
#include<string.h>
void main()
{
    printf("\n\n\tREVERSE A STRING USING POINTERS\n\n");
    char str[100];
    char rev[100];
    char *sptr = str; // sptr stores the base address of the str
    char *rptr = rev; // rptr stores the base address of the reverse
    int i = -1;
    printf("\n\nEnter a string: ");
    gets(str);
    // storing the ending address of str in sptr
    while(*sptr)
    {
        sptr++;
        i++; // i is the index of the end location
    }
}
```

```

}
// storing the string str in rev in reverse order
while(i >= 0)
{
    /* First decrementing then using as it stores
       the location after the end location due to above while loop */
    sptr--;
    *rptr = *sptr; // storing the value in sptr in rptr
    rptr++; // pointing to next location
    i--; // decrementing the index
}
/*
   String should always end with '\0' so explicitly
   putting it at the end of the string
*/
*rptr = '\0';
rptr = rev; // restoring the base address of the reverse string
// storing the reverse string in the original string
while(*rptr)
{
    *sptr = *rptr;
    sptr++;
    rptr++;
}
// printing the reverse string
printf("\n\nReverse of the string is: %s ", str);
    getch();
}

```

OUTPUT



```
C:\Users\HP\Desktop\cprg\stringrevptr.exe

REVERSE A STRING USING POINTERS

Enter a string: GOOD MORNING

Reverse of the string is: GNINROM DOOG
-----
Process exited after 14.89 seconds with return value 13
Press any key to continue . . .
```

3. SEARCH AN ELEMENT IN A 1D ARRAY

```
#include<stdio.h>

void main()
{
    int a[30], ele, num, i,flag=0;
    printf("\nEnter no of elements :");
    scanf("%d", &num);
    printf("\nEnter the values :");
    for (i = 0; i < num; i++)
    {
        scanf("%d", &a[i]);
    }
    printf("\nEnter the elements to be searched :");
    scanf("%d", &ele);
    for(i=0;i<num;i++)
    {
        if(a[i]==ele)
        {
            flag=1;
            break;
        }
    }
}
```

```

        }
    }
    if(flag==1)
        printf("%d is found at position %d",ele,i+1);
    else
        printf("%d is not found in the array ",ele);
    getch();
}

```

OUTPUT

```

C:\Users\HP\Desktop\cprg\1d.exe
Enter no of elements :7
Enter the values :10 4 56 23 90 3 7
Enter the elements to be searched :23
23 is found at position 4
-----
Process exited after 28.73 seconds with return value 13
Press any key to continue . . .

```

4. SEARCH AN ELEMENT IN AN 2D ARRAY

```

#include<stdio.h>
main()
{
    int array[10][10];
    int i, j, r, c, sum = 0, flag=0, item;
    printf("Enter the order of the matrix\n");
    scanf("%d %d", &r, &c);
    printf("Enter the co-efficients of the matrix\n");
    for (i = 0; i < r; ++i)
    {
        for (j = 0; j < c; ++j)
        {
            scanf("%d", &array[i][j]);
        }
    }
}

```

```

    }
printf("the matrix is \n");
for (i = 0; i < r; ++i)
{
    for (j = 0; j < c; ++j)
    {
        printf("%d\t", array[i][j]);
    }
    printf("\n");
}

printf("Enter theELEMENT TO SEARCH\n");
scanf("%d",&item);
for (i = 0; i < r; ++i)
{
    for (j = 0; j < c; ++j)
    {
        if(array[i][j] ==item)
        {
            flag=1;
            printf("%d is found at row %d and column %d",item,i,j);
        }
    }
}

if(flag==0)
printf("%d is not found in the 2D array",item);
}

```

OUTPUT

```
C:\Users\HP\Desktop\cprg\2D.exe
Enter the order of the matrix
3 4
Enter the co-efficients of the matrix
1 2 3 4 5 6 7 8 9 10 11 12
the matrix is
1      2      3      4
5      6      7      8
9      10     11     12
Enter the element to search
10
10 is found at row 3 and column 2
-----
Process exited after 23.64 seconds with return value 3
Press any key to continue . . .
```

5. IMPLEMENT PASCALS TRIANGLE USING 2D ARRAY

```
#include<stdio.h>

void printPascalTr(int size);

void main()
{
    int size;

    printf("\n\n\tEnter Pascal triangle size: ");
    scanf("%d",&size);
    printPascalTr(size);
    getch();
}

void printPascalTr(int size)
{
    int PascalTr[size][size];
    int row,col;

    //assign zero to every array element
    for(row=0;row<size;row++)
        for(col=0;col<size;col++)
            PascalTr[row][col]=0;

    //first and second rows are set to 1s
    PascalTr[0][0]=1;
    PascalTr[1][0]=1;
    PascalTr[1][1]=1;
    for(row=2;row<size;row++)
    {
```

```

PascalTr[row][0]=1;
for(col=1;col<=row;col++)
{
    PascalTr[row][col]=PascalTr[row-1][col-1]+PascalTr[row-1][col];
}
}
printf("\n\tPASCALS TRINGLE WITH %d ROWS  : \n\n",row);
for(row=0;row<size;row++)
{
    for(col=0;col<=row;col++)
    {
        printf("\t%d",PascalTr[row][col]);
    }
    printf("\n");
}
}

```

OUTPUT

```

C:\Users\HP\Desktop\cprg\PASCAL.exe
Enter Pascal triangle size: 6
PASCALS TRINGLE WITH 6 ROWS  :
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1
-----
Process exited after 5.446 seconds with return value 13
Press any key to continue . . .

```

6. MERGE 2 SORTED ARRAYS INTO 1 SORTED ARRAY

```

#include <stdio.h>

int main()
{
    int array1[50], array2[50], array3[100], m, n, i, j, k = 0;
    printf("\n Enter size of array Array 1: ");
}

```



```

scanf("%d", &m);
printf("\n Enter sorted elements of array 1: \n");
for (i = 0; i < m; i++)
{
    scanf("%d", &array1[i]);
}

printf("\n Enter size of array 2: ");
scanf("%d", &n);
printf("\n Enter sorted elements of array 2: \n");
for (i = 0; i < n; i++)
{
    scanf("%d", &array2[i]);
}
i = 0;
j = 0;
while (i < m && j < n)
{
    if (array1[i] < array2[j])
    {
        array3[k] = array1[i];
        i++;
    }
    else
    {
        array3[k] = array2[j];
        j++;
    }
    k++;
}
if (i >= m)
{

```

```

while (j < n)
{
    array3[k] = array2[j];
    j++;
    k++;
}
}
if (j >= n)
{
    while (i < m)
    {
        array3[k] = array1[i];
        i++;
        k++;
    }
}

printf("\n After merging: \n");
for (i = 0; i < m + n; i++)
{
    printf("\n%d", array3[i]);
}
return 0;
}

```

OUTPUT

```

Enter size of array Array 1: 4
Enter sorted elements of array 1:
1 5 8 10
Enter size of array 2: 5
Enter sorted elements of array 2:
2 3 9 11 15
After merging:
1
2
3
5
8
9
10
11
15
-----
Process exited after 193.8 seconds with return value 0
Press any key to continue . . .

```

7. SEARCH AN ELEMENT IN AN ARRAY USING RECURSIVE BINARY SEARCH

```

#include <stdio.h>
void binary_search(int [], int, int, int);
void main()
{
    int key, n, i;
    int array[100];
    printf("Enter number of elements(between 1 & 100) ");
    scanf("%d", &n);
    printf("\nEnter %d integers in ascending order\n", n);
    for(i = 0; i < n; i++)
    {
        scanf("%d",&array[i]);
    }
    printf("\n");
    printf("Enter value to find\n");
    scanf("%d", &key);
    binary_search(array, 0, n, key);
}
void binary_search(int array[], int low, int high, int key)
{

```

```

int mid;
if (low > high)
{
    printf(" %d isn't present in the list or list is not in ascending order\n", key);
    return;
}
mid = (low + high) / 2;
if (array[mid] == key)
{
    printf("\n%d found at location %d\n", key, mid+1);
}
else if (array[mid] > key)
{
    binary_search(array, low, mid - 1, key);
}
else if (array[mid] < key)
{
    binary_search(array, mid + 1, high, key);
}
}

```

OUTPUT

```

C:\Users\HP\Desktop\cprg\BINREC.exe
Enter number of elements(between 1 & 100) 10
Enter 10 integers in ascending order
1 2 3 4 5 6 7 8 9 10
Enter value to find
4
4 found at location 4
-----
Process exited after 24.7 seconds with return value 23
Press any key to continue . . .

```

8. IMPLEMENT SPARSE MATRIX

```

#include<stdio.h>

void main()
{

```

```

int A[10][10],B[10][3],m,n,s=0,i,j;
printf("\nEnter the order m x n of the sparse matrix\n");
scanf("%d%d",&m,&n);
printf("\nEnter the elements in the sparse matrix(mostly zeroes)\n");
for(i=0;i<m;i++)
{
for(j=0;j<n;j++)
{
printf("\n%d row and %d column: ",i+1,j+1);
scanf("%d",&A[i][j]);
}
}
printf("The given matrix is:\n");
for(i=0;i<m;i++)
{
for(j=0;j<n;j++)
{
printf("%d ",A[i][j]);
}
printf("\n");
}
for(i=0;i<m;i++)
{
for(j=0;j<n;j++)
{
if(A[i][j]!=0)
{
B[s][0]=A[i][j];
B[s][1]=i+1;
B[s][2]=j+1;
s++;
}
}
}

```

```

}
}
printf("\nThe sparse matrix is given by\n");
printf("\n");
printf("\tvalue\trow\tcolumn\n\n");
for(i=0;i<s;i++)
{
for(j=0;j<3;j++)
{
printf("\t%d ",B[i][j]);
}
printf("\n");
}
getch();
}

```

OUTPUT

The screenshot shows a command prompt window titled "C:\Users\HP\Desktop\cprg\SPARSE.exe". The user enters "2 3" for the matrix order. Then, they input the matrix elements row by row: (1,1)=1, (1,2)=0, (1,3)=0, (2,1)=0, (2,2)=7, (2,3)=0. The program then displays the matrix and a table of non-zero elements.

```

C:\Users\HP\Desktop\cprg\SPARSE.exe
Enter the order m x n of the sparse matrix
2 3
Enter the elements in the sparse matrix(mostly zeroes)
1 row and 1 column: 1
1 row and 2 column: 0
1 row and 3 column: 0
2 row and 1 column: 0
2 row and 2 column: 7
2 row and 3 column: 0
The given matrix is:
1 0 0
0 7 0
The sparse matrix is given by
    value  row  column
    1      1    1
    7      2    2

```

9. IMPLEMENT POLYNOMIAL USING ARRAYS

```

#include<stdio.h>

void main()
{

```

```

int c[10];
int o,i;
printf("Enter the order of polynomial : ");
scanf("%d",&o);
printf("\nEnter the coefficients of polynomial\n\n");
for(i=0;i<=o;i++)
{
    printf("coefficient of X^%d : ",i);
    scanf("%d",&c[i]);
}

printf("\nentered polynomial : ");
for(i=o;i>0;i--)
{
    if(c[i]!=0)

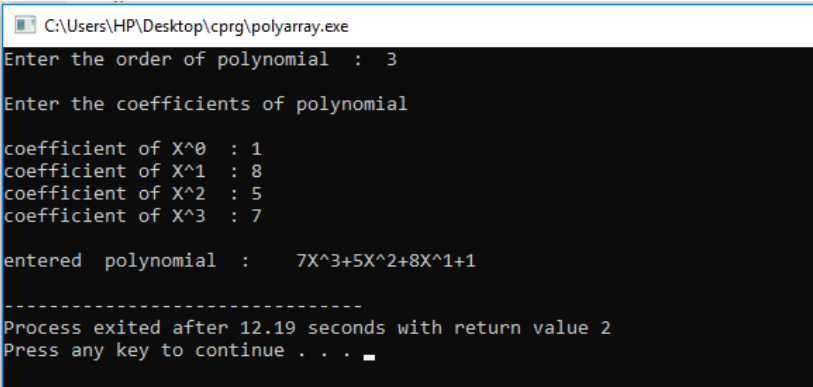
        printf("%dX^%d+",c[i],i);

}
printf("%d\n",c[i]);

getch();
}

```

OUTPUT



```

C:\Users\HP\Desktop\cprg\polyarray.exe
Enter the order of polynomial : 3
Enter the coefficients of polynomial
coefficient of X^0 : 1
coefficient of X^1 : 8
coefficient of X^2 : 5
coefficient of X^3 : 7
entered polynomial : 7X^3+5X^2+8X^1+1
-----
Process exited after 12.19 seconds with return value 2
Press any key to continue . . .

```

10. IMPLEMENT SINGLY LINKED LIST OF INTEGERS

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

```

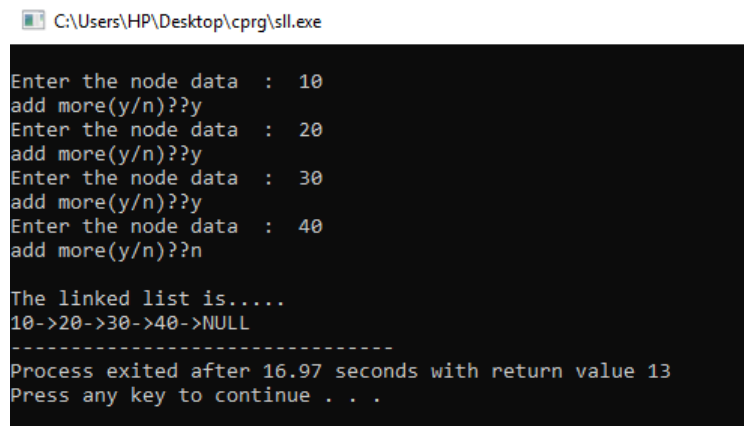
#include<stdlib.h>
void main()
{
    struct node
    {
        int data;
        struct node *link;
    }*start=NULL,*new_node,*ptr;
char ch;
    do
    {
        new_node=(struct node*)malloc(sizeof(struct node));
        printf("\nEnter the node data : ");
        scanf("%d",&new_node->data);
        new_node->link=NULL;
        if(start==NULL)
        {
            start=new_node;
            ptr=new_node;
        }
        else
        {
            ptr->link=new_node;
            ptr=new_node;
        }
        printf("add more(y/n)??");
        ch=getche();
    }while(ch=='y'||ch=='Y');
    printf("\n\nThe linked list is.....\n");
    for(ptr=start;ptr!=NULL;ptr=ptr->link)
        printf("%d->",ptr->data);
    printf("NULL");
}

```



```
    getch();  
}
```

OUTPUT



```
C:\Users\HP\Desktop\cprg\sl.exe  
Enter the node data : 10  
add more(y/n)?y  
Enter the node data : 20  
add more(y/n)?y  
Enter the node data : 30  
add more(y/n)?y  
Enter the node data : 40  
add more(y/n)?n  
  
The linked list is....  
10->20->30->40->NULL  
-----  
Process exited after 16.97 seconds with return value 13  
Press any key to continue . . .
```

11. a) DELETION OF A ELEMENT FROM SINGLY LINKED LIST(Based on item)

```
#include<stdio.h>  
#include<stdlib.h>  
void main()  
{  
    struct node  
    {  
        int data;  
        struct node*link;  
    }*start=NULL,*new_node,*ptr;  
    char ch;  
    do  
    {  
        new_node=(struct node*)malloc(sizeof (struct node));  
        printf("\nenter the node data:");  
        scanf("%d",&new_node->data);  
        new_node->link=NULL;  
        if(start==NULL)  
        {  
            start=new_node;  
            ptr=new_node;  
        }  
    }  
}
```

```

    }
    else
    {
        ptr->link=new_node;
        ptr=new_node;
    }
    printf("add more(y/n)??");
    ch=getche();
}while(ch=='y'||ch=='Y');
printf("\n\nthe linked list is ..... \n");
for(ptr=start;ptr!=NULL;ptr=ptr->link)
printf("%d->",ptr->data);
printf("NULL");
struct node *prev;
int item;
char ch1;
do
{
    if(start==NULL)
    {
        printf("\nmemory underflow-list is empty");
        exit(0);
    }
    else
    {
        printf("\nenter the item for deletion:");
        scanf("%d",&item);
        ptr=start;
        while(item!=ptr->data&&ptr!=NULL)
        {
            prev=ptr;
            ptr=ptr->link;

```

```

    }
    if(ptr==NULL)
    printf("\nLOCATION ENTERED NOT EXIST");
    else if(ptr==start)
    {
        start=ptr->link;
        free(ptr);
    }
    else
    {
        prev->link=ptr->link;
        free(ptr);
    }
    printf("\n\nTHE LINKED LIST AFTER DELETION.....\n");
    for(ptr=start;ptr!=NULL;ptr=ptr->link)
    printf("%d->",ptr->data);
    printf("NULL");
}
printf("\ndelete more(y/n)??");
ch1=getche();
}while(ch1=='y'||ch1=='Y');
getch();
}

```

OUTPUT

```

C:\Users\HP\Desktop\cprg\deletion.exe
enter the node data:10
add more(y/n)?Y
enter the node data:20
add more(y/n)?Y
enter the node data:30
add more(y/n)?Y
enter the node data:40
add more(y/n)?N

the linked list is .....
10->20->30->40->NULL
enter the item for deletion:30

THE LINKED LIST AFTER DELETION.....
10->20->40->NULL
delete more(y/n)?N
-----
Process exited after 29.4 seconds with return value 13
Press any key to continue . . .

```

b) DELETION OF A ELEMENT FROM SINGLY LINKED LIST(Based on node position)

```

#include<stdio.h>

#include<stdlib.h>

void main()
{
    struct node
    {
        int data;
        struct node *link;
    }*start=NULL,*new_node,*ptr;
char ch;
    do
    {
        new_node=(struct node*)malloc(sizeof(struct node));
        printf("\nEnter the node data : ");
        scanf("%d",&new_node->data);
        new_node->link=NULL;
        if(start==NULL)
        {
            start=new_node;
            ptr=new_node;
        }
        else

```

```

        {
            ptr->link=new_node;
            ptr=new_node;
        }
        printf("add more(y/n)??");
        ch=getche();
    }while(ch=='y'||ch=='Y');
    printf("\n\nThe linked list is.....\n");
    for(ptr=start;ptr!=NULL;ptr=ptr->link)
        printf("%d->",ptr->data);
    printf("NULL");
    struct node *prev;
    int loc;
    char ch1;
    do
    {
        if(start==NULL)
        {
            printf("\nMemory underflow-list is empty");
            exit(0);
        }
        else
        {
            printf("\nEnter the location or node number for deletion :");
            scanf("%d",&loc);
            ptr=start;
            int i=1;
            while(i<loc&&ptr!=NULL)
            {
                prev=ptr;
                ptr=ptr->link;
                i++;
            }
        }
    }

```

```

    }
    if(ptr==NULL||loc<=0)
    printf("\nLOCATION ENTERED NOT EXIST");
    else if(ptr==start)
    {
        start=ptr->link;
        free(ptr);
    }
    else
    {
        prev->link=ptr->link;
        free(ptr);
    }
    printf("\n\nTHE LINKED LIST AFTER DELETION.....\n");
    for(ptr=start;ptr!=NULL;ptr=ptr->link)
    printf("%d->",ptr->data);
    printf("NULL");
}
printf("\nDelete more(y/n)??");
    ch1=getche();
}while(ch1=='y'||ch1=='Y');
getch();
}

```

OUTPUT

```

C:\Users\HP\Desktop\cprg\LLDELP.exe
Enter the node data : 10
add more(y/n)??Y
Enter the node data : 20
add more(y/n)??Y
Enter the node data : 30
add more(y/n)??Y
Enter the node data : 40
add more(y/n)??N

The linked list is....
10->20->30->40->NULL
Enter the location or node number for deletion :3

THE LINKED LIST AFTER DELETION....
10->20->40->NULL
Delete more(y/n)??N
-----
Process exited after 25.49 seconds with return value 13
Press any key to continue . . .

```

12. IMPLEMENT A DOUBLY LINKED LIST OF INTEGERS

```

#include<stdio.h>
#include<stdlib.h>
void main()
{
    struct node
    {
        int data;
        struct node *link,*prev;
    }*start=NULL,*new_node,*ptr;
printf("\n\n\tDOUBLY LINKED LIST\n\n");
char ch;
do
{
    new_node=(struct node *)malloc(sizeof(struct node));
    printf("\n enter the node data:");
    scanf("%d",&new_node->data);
    new_node->link=NULL;
    if(start==NULL)
    {
        new_node->prev=NULL;

```

```

        start=new_node;

        ptr=new_node;
    }
    else
    {
        ptr->link=new_node;
        new_node->prev=new_node;
        ptr=new_node;
    }

    printf("add more(y/n)??");

    ch=getche();
}while(ch=='Y'||ch=='y');
printf("\n\nthe Doubly Linked List is \n");
for(ptr=start;ptr!=NULL;ptr=ptr->link)
printf("%d<->",ptr->data);
printf("NULL");
}

```

OUTPUT

```

C:\Users\HP\Desktop\cprg\student\dll.exe

DOUBLY LINKED LIST

enter the node data:10
add more(y/n)??Y
enter the node data:20
add more(y/n)??Y
enter the node data:30
add more(y/n)??Y
enter the node data:40
add more(y/n)??N

the Doubly Linked List is
10<->20<->30<->40<->NULL
-----
Process exited after 16.48 seconds with return value 4
Press any key to continue . . .

```

13. IMPLEMENTATION OF CIRCULAR LINKED LIST

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

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

```
void main()
```



```

{
    struct node
    {
        int data;
        struct node *link;
    }*start=NULL,*new_node,*ptr;
char ch;
do
{
    new_node=(struct node*)malloc(sizeof(struct node));
    printf("\nEnter the node data : ");
    scanf("%d",&new_node->data);
    new_node->link=start;
    if(start==NULL)
    {
        start=new_node;
        ptr=new_node;
    }
    else
    {
        ptr->link=new_node;
        ptr=new_node;
    }
    printf("add more(y/n)??");
    ch=getche();
}while(ch=='y'||ch=='Y');
printf("\n\nThe linked list is.....\n");
ptr=start;
do
{
    printf("%d->",ptr->data);
    ptr=ptr->link;
}

```

```

    }while(ptr!=start);

    printf("LINK BACK TO FIRST NODE");

    getch();
}

```

OUTPUT

```

C:\Users\HP\Desktop\cprg\CL.exe
Enter the node data : 5
add more(y/n)?Y
Enter the node data : 1
add more(y/n)?Y
Enter the node data : 9
add more(y/n)?Y
Enter the node data : 4
add more(y/n)?N

The linked list is....
5->1->9->4->LINK BACK TO FIRST NODE_

```

14. IMPLEMENT POLYNOMIAL USING LINKED LIST

```

#include<stdio.h>
#include<stdlib.h>
struct link
{
    int coeff;
    int pow;
    struct link *next;
}*poly=NULL;
void create(struct link *node)
{
    char ch;
    do
    {
        printf("\nenter coeff : ");
        scanf("%d",&node->coeff);
        printf("\n enter power : ");
        scanf("%d",&node->pow);
        node->next=(struct link*)malloc(sizeof(struct link));
    }
}

```

```

node=node->next;
node->next=NULL;
printf("\n continue(y/n) : ");
ch=getche();
}
while(ch=='y' || ch=='Y');
}
void show(struct link *node)
{
while(node->next!=NULL)
{
printf("%dx^%d",node->coeff,node->pow);
node=node->next;
if(node->next!=NULL)
printf("+");
}
}
void main()
{
poly=(struct link *)malloc(sizeof(struct link));
printf("\nEnter polynomial : \n\n");
create(poly);
printf("\n\nThe Polynomial is : ");
show(poly);
getch();
}

```

OUTPUT

```
C:\Users\HP\Desktop\cprg\polyll.exe
Enter polynomial :
enter coeff : 2
enter power : 3
continue(y/n) : Y
enter coeff : 3
enter power : 2
continue(y/n) : Y
enter coeff : 6
enter power : 1
continue(y/n) : N
The Polynomial is : 2x^3+3x^2+6x^1
-----
Process exited after 24.91 seconds with return value 0
Press any key to continue . . .
```

15. ADDITION OF 2 POLYNOMIALS

```
#include<stdio.h>
void main()
{
    int poly1[10],poly2[10],poly[10];
    int or1,or2,i;
    printf("Enter the order of first polynomial : ");
    scanf("%d",&or1);
    printf("\nEnter the coefficients of first polynomial\n\n");
    for(i=0;i<=or1;i++)
    {
        printf("coefficient of X^%d : ",i);
        scanf("%d",&poly1[i]);
    }
    printf("Enter the order of second polynomial : ");
    scanf("%d",&or2);
    printf("\nEnter the coefficients of second polynomial\n\n");
    for(i=0;i<=or2;i++)
    {
        printf("coefficient of X^%d : ",i);
        scanf("%d",&poly2[i]);
    }
}
```

```

printf("\nFirst polynomial : ");
for(i=or1;i>0;i--)
{
    if(poly1[i]!=0)

        printf("%dX^%d+",poly1[i],i);

}
printf("%d\n",poly1[i]);
    printf("\nSecond polynomial : ");
for(i=or2;i>0;i--)
{
    if(poly2[i]!=0)
        printf("%dX^%d+",poly2[i],i);

}
printf("%d\n",poly2[i]);
//Adding the polynomials
if(or1>or2)
{
    for(i=0;i<=or2;i++)
    {
        poly[i]=poly1[i]+poly2[i];
    }
    for(i=or2+1;i<=or1;i++)
    {
        poly[i] = poly1[i];
    } }
else
{
    for(i=0;i<=or1;i++)

```

```

    {
        poly[i]=poly1[i]+poly2[i];
    }
for(i=or1+1;i<=or2;i++)
{
    poly[i] = poly2[i];
}
}

printf("\nPolynomial after addition : ");
for(i=or2+or1;i>0;i--)
{
    if(poly[i]!=0)
        printf("%dX^%d+",poly[i],i);
}
printf("%d\n",poly[i]);
getch();
}

```

OUTPUT

```

C:\Users\HP\Desktop\cprg\polyadd.exe
Enter the order of first polynomial : 3
Enter the coefficients of first polynomial
:
coefficient of X^0 : 2
coefficient of X^1 : 7
coefficient of X^2 : 3
coefficient of X^3 : 4
Enter the order of second polynomial : 2
Enter the coefficients of second polynomial
coefficient of X^0 : 1
coefficient of X^1 : 9
coefficient of X^2 : 6
First polynomial : 4X^3+3X^2+7X^1+2
Second polynomial : 6X^2+9X^1+1
Polynomial after addition : 4X^3+9X^2+16X^1+3
_

```

16. IMPLEMENT STACK USING ARRAY

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

```
int stack[100],choice,n,top,x,i;
```

```

void push();
void pop();
void display();
int main()
{
    top=-1;
    printf("\n Enter the size of STACK[MAX=100]:");
    scanf("%d",&n);
    printf("\n\t STACK OPERATIONS USING ARRAY");
    printf("\n\t-----");
    printf("\n\t 1.PUSH\n\t 2.POP\n\t 3.DISPLAY\n\t 4.EXIT");
    do
    {
        printf("\n Enter the Choice:");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
            {
                push();
                break;
            }
            case 2:
            {
                pop();
                break;
            }
            case 3:
            {
                display();
                break;
            }
        }
    }
}

```

```

    case 4:
    {
        printf("\n\t EXIT POINT ");
        break;
    }
    default:
    {
        printf ("\n\t Please Enter a Valid Choice(1/2/3/4)");
    }

}

}

while(choice!=4);
return 0;
}

void push()
{
    if(top>=n-1)
    {
        printf("\n\tSTACK is over flow");
    }
    else
    {
        printf(" Enter a value to be pushed:");
        scanf("%d",&x);
        top++;
        stack[top]=x;
    }
}

void pop()
{
    if(top<=-1)

```



```

{
    printf("\n\t Stack is under flow");
}
else
{
    printf("\n\t The popped elements is %d",stack[top]);
    top--;
}
}
void display()
{
    if(top>=0)
    {
        printf("\n The elements in STACK \n");
        for(i=top; i>=0; i--)
            printf("\n%d",stack[i]);
        printf("\n Press Next Choice");
    }
    else
    {
        printf("\n The STACK is empty");
    }
}

```

OUTPUT

```

C:\Users\HP\Desktop\cprg\stackusingarray.exe
Enter the size of STACK[MAX=100]:10

    STACK OPERATIONS USING ARRAY
-----
    1.PUSH
    2.POP
    3.DISPLAY
    4.EXIT
Enter the Choice:1
Enter a value to be pushed:10

Enter the Choice:1
Enter a value to be pushed:20

Enter the Choice:1
Enter a value to be pushed:30

Enter the Choice:3

The elements in STACK

30
20
10
Press Next Choice
Enter the Choice:2

    The popped elements is 30
Enter the Choice:3

The elements in STACK

20
10
Press Next Choice
Enter the Choice:4

    EXIT POINT
-----
Process exited after 39.47 seconds with return value 0
Press any key to continue . . .

```

17. IMPLEMENT STACK USING LINKED LIST

```

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

void push();
void pop();
void display();

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

int main ()
{
int choice=0;

printf("\n\t STACK OPERATIONS USING LINKED LIST");
printf("\n\t-----");

```

```

printf("\n\t 1.PUSH\n\t 2.POP\n\t 3.DISPLAY\n\t 4.EXIT");
do
{
    printf("\n Enter the Choice:");
    scanf("%d",&choice);
    switch(choice)
    {
        case 1:
        {
            push();
            break;
        }
        case 2:
        {
            pop();
            break;
        }
        case 3:
        {
            display();
            break;
        }
        case 4:
        {
            printf("\n\t EXIT POINT ");
            break;
        }
        default:
        {
            printf ("\n\t Please Enter a Valid Choice(1/2/3/4)");
        }
    }
}

```

```

    }
}
while(choice!=4);
return 0;
}
void push ()
{
int data;
struct node *newnode = (struct node*)malloc(sizeof(struct node));
    printf("Enter the data");
scanf("%d",&data);
newnode->data = data;
if(start==NULL)
{
    newnode -> next = NULL;
    start=newnode;
}
else
{
    newnode ->next = start;
    start=newnode ;
}
printf("Item pushed");
}
void pop()
{
int item;
struct node *ptr;
if (start == NULL)
{
    printf("Underflow");
}
}

```

```

else
{
    item = start->data;
    ptr = start;
    start = start->next;
    free(ptr);
    printf("Item popped");
}
}
void display()
{
    int i;
    struct node *ptr;
    ptr=start;
    if(ptr == NULL)
    {
        printf("Stack is empty\n");
    }
    else
    {
        printf("Printing Stack elements \n");
        while(ptr!=NULL)
        {
            printf("%d\n",ptr->data);
            ptr = ptr->next;
        }
    }
}

```

OUTPUT

```

C:\Users\HP\Desktop\cprg\stackusingll.exe

STACK OPERATIONS USING LINKED LIST
-----
1.PUSH
2.POP
3.DISPLAY
4.EXIT
Enter the Choice:1
Enter the data 10
Item pushed
Enter the Choice:1
Enter the data 20
Item pushed
Enter the Choice:1
Enter the data 30
Item pushed
Enter the Choice:3
Printing Stack elements
30
20
10

Enter the Choice:2
Item popped
Enter the Choice:3
Printing Stack elements
20
10

Enter the Choice:1
Enter the data50
Item pushed
Enter the Choice:3
Printing Stack elements
50
20
10

Enter the Choice:4

EXIT POINT
-----
Process exited after 49.2 seconds with return value 0
Press any key to continue . . .

```

18. IMPLEMENT QUEUE USING ARRAY

```

#include <stdio.h>

void insert();

void delete();

void display();

int queue_array[100];

int rear = - 1;

int front = - 1;

main()
{
    int choice;

    printf("\n\t QUEUE OPERATIONS USING ARRAY");

    printf("\n\t-----\n");

    printf("1.Insert element to queue \n");

    printf("2.Delete element from queue \n");

```

```

printf("3.Display all elements of queue \n");
printf("4.Quit \n");
do
{
printf("Enter your choice : ");
scanf("%d", &choice);
switch (choice)
{
case 1:
insert();
break;
case 2:
delete();
break;
case 3:
display();
break;
case 4:
printf("\n\t EXIT POINT ");
break;
default:
printf ("\n\t Please Enter a Valid Choice(1/2/3/4)");
}
}while(choice!=4);
}

```

```

void insert()
{
int add_item;
if (rear == 99)
printf("Queue Overflow \n");
else

```

```

{
    if (front == - 1)
        front = 0;
    printf("Enter a value to be inserted : ");
    scanf("%d", &add_item);
    rear = rear + 1;
    queue_array[rear] = add_item;
}
}
void delete()
{
    if (front == - 1 || front > rear)
    {
        printf("Queue Underflow \n");
        return ;
    }
    else
    {
        printf("Element deleted from queue is : %d\n", queue_array[front]);
        front = front + 1;
    }
}
void display()
{
    int i;
    if (front == - 1||front>rear)
        printf("Queue is empty \n");
    else
    {
        printf("Queue is : \n");
        for (i = front; i <= rear; i++)
            printf("%d ", queue_array[i]);
    }
}

```



```

        printf("\n");
    }
}

```

OUTPUT

```

C:\Users\HP\Desktop\cprg\qarray.exe

        QUEUE OPERATIONS USING ARRAY
    -----
1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Quit
Enter your choice : 1
Enter a value to be inserted : 10
Enter your choice : 1
Enter a value to be inserted : 20
Enter your choice : 1
Enter a value to be inserted : 30
Enter your choice : 3
Queue is :
10 20 30
Enter your choice : 2
Element deleted from queue is : 10
Enter your choice : 3
Queue is :
20 30
Enter your choice : 4

        EXIT POINT
    -----
Process exited after 24.18 seconds with return value 4
Press any key to continue . . .

```

19. IMPLEMET QUEUE USING LINKED LIST

```

#include <stdio.h>

#include<stdlib.h>

void insert();

void delete();

void display();

struct node
{
int data;
struct node *next;
}*front=NULL,*rear=NULL;

void main()
{
int choice;

printf("\n\t QUEUE OPERATIONS USING LINKED LIST");

printf("\n\t-----\n");

```

```

printf("1.Insert element to queue \n");
printf("2.Delete element from queue \n");
printf("3.Display all elements of queue \n");
printf("4.Quit \n");
do
{
printf("\nEnter your choice : ");
scanf("%d", &choice);
switch (choice)
{
case 1:
insert();
break;
case 2:
delete();
break;
case 3:
display();
break;
case 4:
printf("\n\t----EXIT POINT---- ");
break;
default:
printf ("\nPlease Enter a Valid Choice(1/2/3/4)");
}
}while(choice!=4);
getch();
}
void insert()
{
int data;
struct node *newnode = (struct node*)malloc(sizeof(struct node));

```

```

printf("Enter the data : ");
scanf("%d",&data);
newnode->data = data;
newnode -> next = NULL;
if(front == NULL)
    front = rear = newnode;
else
{
    rear -> next = newnode;
    rear = newnode;
}
printf("Insertion is Success!!\n");
}
void delete()
{
    if(front == NULL)
        printf("\nQueue is Empty!!!");
    else
    {
        struct node *temp = front;
        front = front -> next;
        printf("Deleted element: %d\n", temp->data);
        free(temp);
    }
}
void display()
{
    if(front == NULL)
        printf("\nQueue is Empty!!!\n");
    else
    {
        struct node *temp = front;

```

```

while(temp->next != NULL)
    {
        printf("%d-->",temp->data);
        temp = temp -> next;
    }
printf("%d-->NULL\n",temp->data);
}
}

```

OUTPUT

```

C:\Users\HP\Desktop\cprg\qll.exe
      QUEUE OPERATIONS USING LINKED LIST
-----
1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Quit

Enter your choice : 1
Enter the data : 10
Insertion is Success!!!

Enter your choice : 1
Enter the data : 20
Insertion is Success!!!

Enter your choice : 1
Enter the data : 30
Insertion is Success!!!

Enter your choice : 3
10-->20-->30-->NULL

Enter your choice : 2
Deleted element: 10

Enter your choice : 3
20-->30-->NULL

Enter your choice : 4

----EXIT POINT----
-----
Process exited after 50.68 seconds with return value 13
Press any key to continue . . .

```

20. BUBBLE SORT

```

#include <stdio.h>

void main()
{
    int n, i, j, temp;
    int array[100];

```

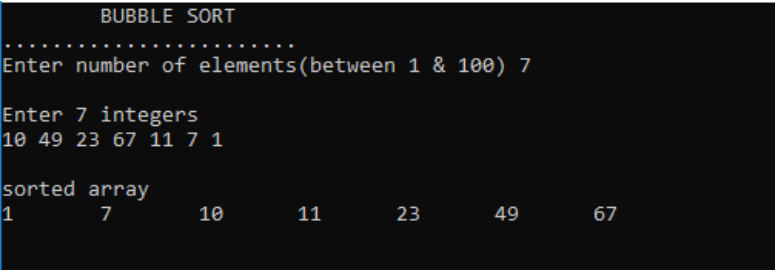
```

printf("\tBUBBLE SORT\n.....\n");
printf("Enter number of elements(between 1 & 100) ");
scanf("%d", &n);
printf("\nEnter %d integers \n", n);
for(i = 0; i < n; i++)
{
    scanf("%d",&array[i]);
}
    for(i=0;i<n-1;i++)
        for(j=0;j<(n-i-1);j++)
            if(array[j]>array[j+1])
                {
                    temp=array[j];
                    array[j]=array[j+1];
                    array[j+1]=temp;
                }
    printf("\nsorted array\n");
    for(i = 0; i < n; i++)
        printf("%d\t",array[i]);

getch();
}

```

OUTPUT



```

BUBBLE SORT
.....
Enter number of elements(between 1 & 100) 7

Enter 7 integers
10 49 23 67 11 7 1

sorted array
1      7      10      11      23      49      67

```

21. INSERTION SORT

```

#include<stdio.h>

void main()

{

```

```

int n, i, j, temp, array[100];

printf("\nINSERTION SORT\n\n");

printf("Enter the size of the list: ");

scanf("%d", &n);

printf("Enter %d integer values: ", n);

for (i = 0; i < n; i++)

    scanf("%d", &array[i]);

//Insertion sort logic

for (i = 1; i < n; i++)

{

    temp = array[i];

    j = i - 1;

    while ((temp < array[j]) && (j >= 0))

    {

        array[j + 1] = array[j];

        j = j - 1;

    }

    array[j + 1] = temp;

}

printf("List after Sorting is: ");

for (i = 0; i < n; i++)

    printf(" %d", array[i]);

getch();

}

```

OUTPUT

```
C:\Users\HP\Desktop\cprg\ins.exe
INSERTION SORT
Enter the size of the list: 15
Enter 15 integer values: 36 87 1 0 33 8 3 2 10 99 9 30 6 4 100
List after Sorting is: 0 1 2 3 4 6 8 9 10 30 33 36 87 99 100
-----
Process exited after 97.85 seconds with return value 13
Press any key to continue . . .
```

22. SELECTION SORT

```
#include<stdio.h>

void main()

{

    int n,i,j,temp,array[100];

    printf("\nSELECTION SORT\n\n");

    printf("Enter the size of the List: ");

    scanf("%d",&n);

    printf("Enter %d integer values: ",n);

    for(i=0; i<n; i++)

        scanf("%d",&array[i]);

    //Selection sort logic

    for(i=0; i<n; i++)

    {

        for(j=i+1; j<n; j++)

            {

                if(array[i] > array[j])

                    {

                        temp=array[i];

                        array[i]=array[j];

                        array[j]=temp;

                    }

            }

    }

}
```

```

        }
    }
}

printf("List after sorting is: ");

for(i=0; i<n; i++)

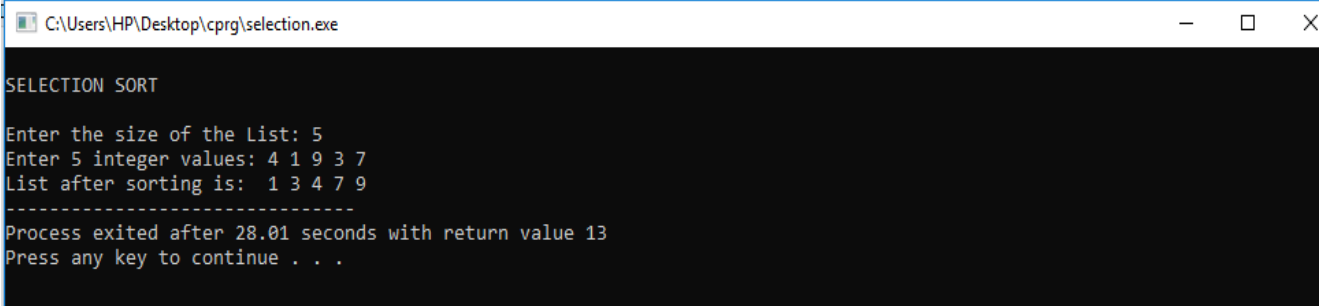
    printf(" %d",array[i]);

getch();

}

```

OUTPUT



```

C:\Users\HP\Desktop\cprg\selection.exe
SELECTION SORT
Enter the size of the List: 5
Enter 5 integer values: 4 1 9 3 7
List after sorting is: 1 3 4 7 9
-----
Process exited after 28.01 seconds with return value 13
Press any key to continue . . .

```

23. MERGE SORT

```

#include <stdio.h>

void merge(int a[], int p, int q, int r)

{

    int b[100];

    int i, j, k;

    k = 0;

    i = p;

    j = q + 1;

    while(i <= q && j <= r)

    {

        if(a[i] < a[j])

```



```
{  
    b[k]=a[i];  
        k++;  
        i++;  
}  
else  
{  
    b[k]=a[j];  
        k++;  
        j++;  
}  
}
```

```
while(i <= q)
```

```
{  
    b[k]=a[i];  
        k++;  
        i++;  
}
```

```
while(j <= r)
```

```
{  
    b[k]=a[j];  
        k++;  
j++;  
}
```

```

for(i=r; i >= p; i--)
    {
        --k;
        a[i] = b[k];
    }
}

void mergeSort(int a[], int p, int r)
{
    int q;
    if(p < r)
    {
        q = (p + r) / 2;
        mergeSort(a, p, q);
        mergeSort(a, q+1, r);
        merge(a, p, q, r);
    }
}

void main()
{
    int n,arr[100],i;
    printf("\nMERGE SORT\n\n");
    printf("Enter the size of the List: ");
    scanf("%d",&n);
    printf("Enter %d integer values: ",n);
    for(i=0; i<n; i++)

```

```

scanf("%d",&arr[i]);

mergeSort(arr, 0, n-1);

printf("\nSorted array: \n");

for(i=0; i<n; i++)

printf("%d\t",arr[i]);

getch();

}

```

OUTPUT

```

MERGE SORT
Enter the size of the List: 10
Enter 10 integer values: 2 7 1 0 8 7 9 3 2 6
Sorted array:
0      1      2      2      3      6      7      7      8      9
-----
Process exited after 41.09 seconds with return value 13
Press any key to continue . . . -

```

24. QUICK SORT

```

#include<stdio.h>

void quickSort(int [],int,int);

int main()

{

int list[100],size,i;

printf("\nQUICK SORT\n\n");

printf("Enter size of the list: ");

scanf("%d",&size);

printf("Enter %d integer values: ",size);

for(i = 0; i < size; i++)

scanf("%d",&list[i]);

```

```

quickSort(list,0,size-1);

printf("List after sorting is: ");

for(i = 0; i < size; i++)

    printf(" %d",list[i]);

return 0;

}

void quickSort(int list[],int first,int last)

{

    int pivot,i,j,temp;

    if(first < last)

    {

        pivot = first;

        i = first;

        j = last;

        while(i < j)

        {

            while(list[i] <= list[pivot] && i < last)

                i++;

            while(list[j] > list[pivot])

                j--;

            if(i < j)

            {

                temp = list[i];

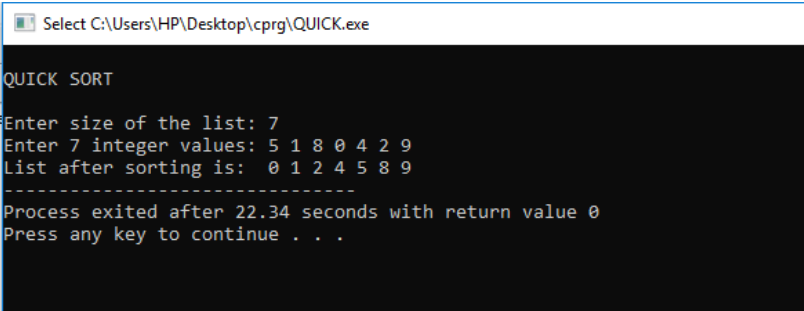
                list[i] = list[j];

                list[j] = temp;

```

```
    }  
}  
  
temp = list[pivot];  
list[pivot] = list[j];  
list[j] = temp;  
quickSort(list,first,j-1);  
quickSort(list,j+1,last);  
}  
}
```

OUTPUT:



```
Select C:\Users\HP\Desktop\cprg\QUICK.exe  
QUICK SORT  
Enter size of the list: 7  
Enter 7 integer values: 5 1 8 0 4 2 9  
List after sorting is: 0 1 2 4 5 8 9  
-----  
Process exited after 22.34 seconds with return value 0  
Press any key to continue . . .
```