|  |
| --- |
| **Program#1 Date:05/08/2022** |
| Write programs to demonstrate the usage of storage classes in C |

**Source Code:**

#include<stdio.h>

int sum=0,a[10]; //Global Variable

void disp(int size,int i)

{

static int index=0; //static variable

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

{

printf("\nIndex =%d\t",index++);

printf("\nValue =%d\t",a[i]);

sum=sum+a[i];

}

printf("\nSum is : %d",sum);

}

void main()

{

int size; //Local Variable

register int i; //Register variable

printf("\n\t\tStorage Classes");

printf("\nEnter the size of Array");

scanf("%d",&size);

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

{

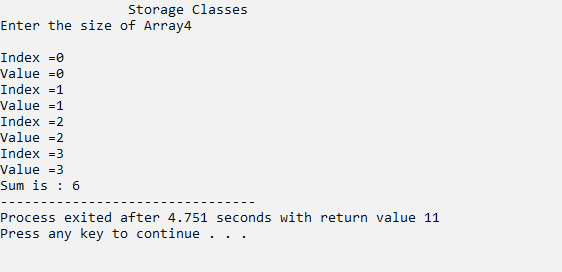
a[i]=i;

}

disp(size,i);

}

**Output:**

****

|  |
| --- |
| **Program#2 Date:05/08/2022** |
| Allocate two-dimensional array dynamically |

**CODE:**

## #include <stdio.h>

## #include <stdlib.h>

## int main()

## {

## int r,c;

## int \*arr;

## int i, j, count = 0;

## printf("Enter the no of rows and columns:");

## scanf("%d%d",&r,&c);

## arr = (int \*)malloc(r \* c \* sizeof(int));

## for (i = 0; i < r; i++)

## {

## for (j = 0; j < c; j++)

## {

## \*(arr + i\*c + j) = ++count;

## }

## }

## for (i = 0; i < r; i++)

## {

## for (j = 0; j < c; j++)

## {

## printf("%d\t", \*(arr + i\*c + j));

## }

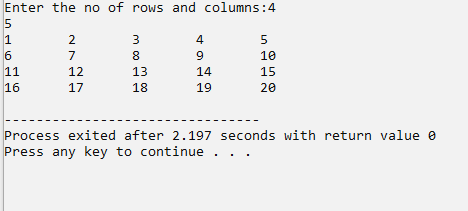
## printf("\n");

## }

## return 0;

## }

**OUTPUT**



|  |
| --- |
| **Program#3 Date:05/08/2022** |
| 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>

int a[10],n=-1;

int readArray()

{

int i;

printf ("Enter the Limit:");

scanf("%d", &n);

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

{

printf (" Enter a[%d]:", i);

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

}

}

int dispArray ()

{

int i;

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

{

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

}

}

int searchElement()

{

int i,s;

printf ("Enter the searching Element:");

scanf ("%d",&s);

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

{

if (a[i]==s)

{

printf("Elementpresent in %d",s,i);

}

}

}

int deleteElement(int p)

{

int i, j;

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

{

if (a[i]==p)

{

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

{

a[j]=a[j+1];

}

n--;

}

}

}

int sortArray()

{

int i,j,temp;

printf ("\n After Sorting;\n");

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

{

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

{

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

{

temp=a[i];

a[i]=a[j];

a[j]=temp;

}

}

}

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

{

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

}

}

int menu()

{

int ch;

printf("\n1. Insertion\n2.Display\n3. Search\n4. Delete\n5. Sort\n6. Exit\n Your Choice:");

scanf ("%d",&ch);

return ch;

}

int main ()

{

int i,s,ch,p,k;

for (ch=menu();ch!=6;ch=menu())

{

switch(ch)

{

case 1:

readArray();

break;

case 2:

dispArray();

break;

case 3:

searchElement();

break;

case 4:

printf("Enter the deleting Element:");

scanf("%d", &p);

deleteElement(p);

break;

case 5:

sortArray();

break;

case 6:

break;

default:

printf("Wrong Choice");

break;

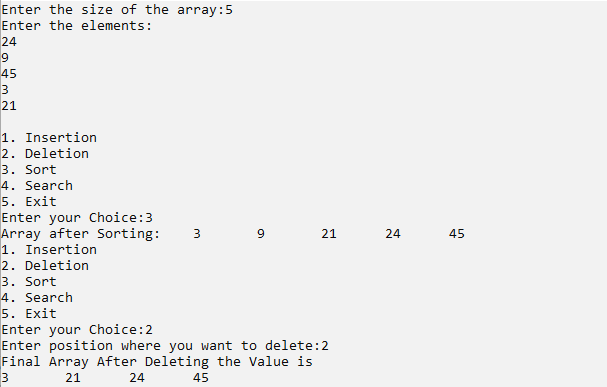
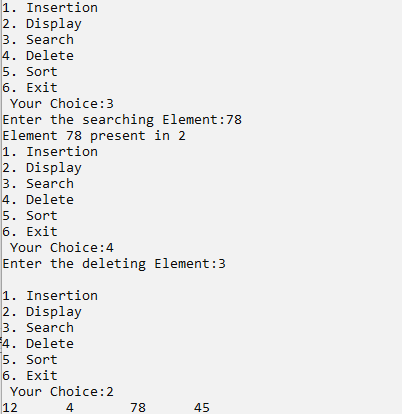
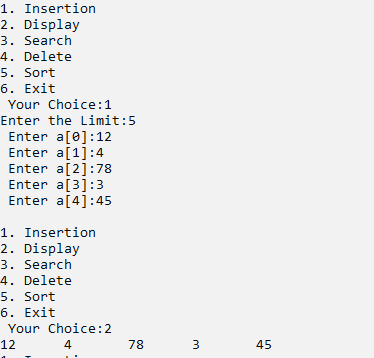
}

}

return 0;

}

**Output:**

****

|  |
| --- |
| **Program#4 Date:05/08/2022** |
| Use a menu driven program to insert, search, delete and sort elements in an array using functions  (use local variables) |

**Source Code:**

#include <stdio.h>

void insert (int \*n, int a[])

{

int value, pos, i;

printf ("Enter the position where you want to insert;");

scanf ("%d", &pos);

printf ("Enter the value into that position:");

scanf ("%d", &value);

for (i=\*n-1; i>=pos-1; i--)

{

a[i+1] = a[i];

}

a[pos-1] = value;

printf ("Final array after inserting the value is\n");

\*n = \*n+1;

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

{

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

}

}

void deleteArray (int \*n, int a[])

{

int pos, i, j;

printf (" Enter position where you want to delete:");

scanf ("%d", &pos);

for (i=pos-1; i<\*n; i++)

{

a[i] = a[i+1];

}

\*n = \*n-1;

printf ("Final Array After Deleting the Value is\n");

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

{

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

}

}

void sort (int n, int a[])

{

int i, j, temp;

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

{

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

{

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

{

temp = a[i];

a[i] = a[j];

a[j] = temp;

}

}

}

printf ("Array after Sorting:\t");

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

{

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

}

}

void search (int n, int a[])

{

int key, i, flag=0;

printf ("Enter the search element:");

scanf ("%d", &key);

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

{

if (a[i] == key)

{

flag = 1;

printf ("Elements Found at Position %d\n", i+1);

}

}

if (flag == 0)

{

printf ("Elements Not Found");

}

}

int menu ()

{

int ch;

printf ("\n1. Insertion\n2. Deletion\n3. Sort\n4. Search\n5. Exit\nEnter your Choice:");

scanf ("%d", &ch);

return ch;

}

void main ()

{

static int n, a[10];

int i;

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

scanf ("%d", &n);

printf ("Enter the elements:");

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

{

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

}

int ch;

for (ch=menu(); ch!=5; ch=menu())

{

switch (ch)

{

case 1:

insert (&n,a);

break;

case 2:

deleteArray (&n,a);

break;

case 3:

sort (n, a);

break;

case 4:

search (n,a);

break;

case 5:

break;

default:

printf ("Wrong Choice");

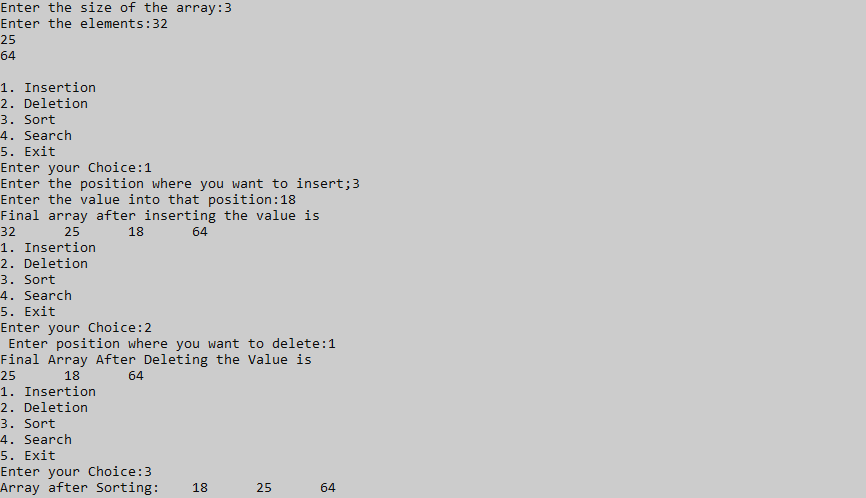
break;

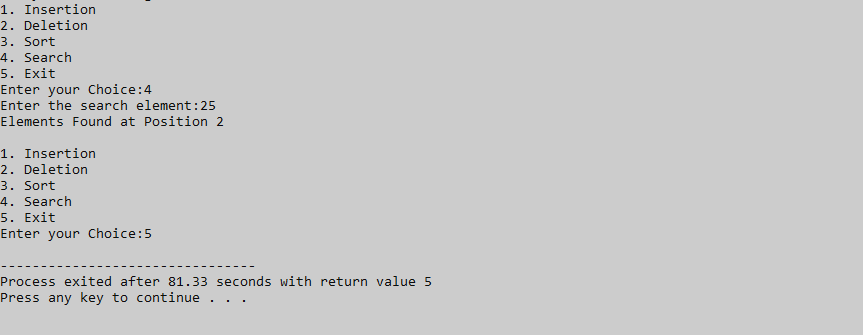
}

}

}

**Output:**





|  |
| --- |
| **Program#5 Date:05/08/2022** |
| Search for all the occurrence of an element in an integer array (positions) |

**Source Code:**

#include<stdio.h>

void main()

{

int a[10],b[10],i,j,n,c=0;

printf("enter the size of the array\n");

scanf("%d",&n);

printf("enter the elements of the array\n");

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

{

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

}

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

{

c=1;

if(a[i]!=-1)

{

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

{

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

{

c++;

a[j]=-1;

}

}

b[i]=c;

}

}

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

{

if(a[i]!=-1)

{

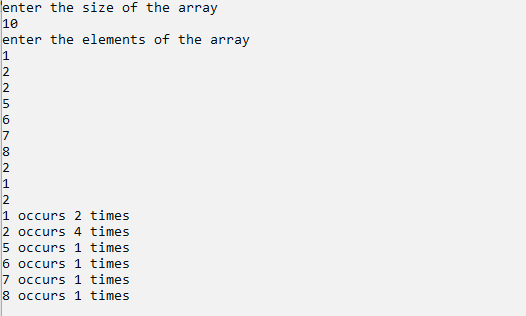
printf("%d occurs %d times\n",a[i],b[i]);

}

}

}

**Output**



|  |
| --- |
| **Program#6 Date:05/08/2022** |
| Sort the array elements in ascending order (minimum three functions - read, disp and sort) |

**Source Code**

#include<stdio.h>

int a[10],n;

int readArray()

{

int i;

printf("Enter the Limit:");

scanf("%d",&n);

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

{

printf("Enter a[%d]:",i);

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

}

}

int dispArray()

{

int i;

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

{

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

}

}

int sortArray()

{

int i,j,temp;

printf("\n After Sorting; \n");

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

{

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

{

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

{

temp=a[i];

a[i]=a[j];

a[j]=temp;

}

}

}

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

{

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

}

}

int main()

{

readArray();

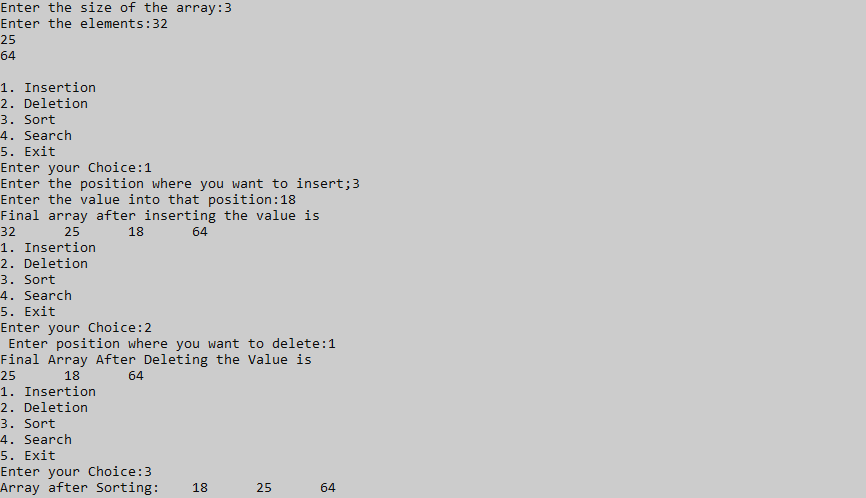
dispArray();

sortArray();

return 0;

}

**Output:**



|  |
| --- |
| **Program#7 Date:17/08/2022** |
| Two Dimensional Matrix - using functions   * 1. Addition   2. Subtraction   3. Multiplication   4. Transpose   5. Determinant |

**Source Code:**

#include<stdio.h> int main()

{

int Array1[20][20],Array2[20][20]; int row,col,count,i,j;

printf("Enter the no of rows and columns of the two matrices:"); scanf("%d %d",&row,&col);

printf("\nEnter the elements of the 1st matrix\n"); count=0;

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

{

for(j=0;j<col;j++)

{

count++;

printf("\nEnter the elements of the a[%d]:",count); scanf("%d",&Array1[i][j]);

}

}

printf("\nEnter the elements of 2nd matrix\n"); count=0;

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

{

for(j=0;j<col;j++)

{

count++;

printf("\nEnter the elements of the a[%d]:",count); scanf("%d",&Array2[i][j]);

}

}

printf("\nArray 1\n"); display(Array1,row,col); printf("\nArray 2\n"); display(Array2,row,col); multiplication(Array1,Array2,row,col); printf("\nTranspose of Array 1\n"); printf("\n"); transpose(Array1,row,col); printf("\nTranspose of Array 2\n"); transpose(Array2,row,col); printf("\nDeterminant of Array 1\n"); determinant(Array1,row);

printf("\n");

printf("\nDeterminant of Array 2\n"); determinant(Array2,row);

return 0;

}

int display(int A[20][20],int row,int col)

{

int i,j; for(i=0;i<row;i++)

{

for(j=0;j<col;j++)

{

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

}

printf("\n");

}

}

int multiplication(int A[20][20],int B[20][20],int row,int col) //multiplication

{

int i,j,k,mul[20][20]; for(i=0;i<row;i++)

{

for(j=0;j<col;j++)

{

mul[i][j]=0; for(k=0;k<col;k++)

{

mul[i][j]=mul[i][j]+A[i][k]\*B[k][j];

}

}

}

printf("\nProduct of two matrices is :\n"); for(i=0;i<row;i++)

{

for(j=0;j<col;j++)

{

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

}

printf("\n");

}

}

int transpose(int a[20][20], int r, int c) //transpose

{

int i,j;

int t[100][100];

{

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

{

for(j=0;j<c;j++)

{

t[j][i]=a[i][j];

}

}

int display(int A[20][20],int row,int col)

{

int i,j; for(i=0;i<row;i++)

{

for(j=0;j<col;j++)

{

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

}

printf("\n");

}

}

int multiplication(int A[20][20],int B[20][20],int row,int col) //multiplication

{

int i,j,k,mul[20][20]; for(i=0;i<row;i++)

{

for(j=0;j<col;j++)

{

mul[i][j]=0; for(k=0;k<col;k++)

{

mul[i][j]=mul[i][j]+A[i][k]\*B[k][j];

}

}

}

printf("\nProduct of two matrices is :\n"); for(i=0;i<row;i++)

{

for(j=0;j<col;j++)

{

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

}

printf("\n");

}

}

int transpose(int a[20][20], int r, int c) //transpose

{

int i,j;

int t[100][100];

{

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

{

for(j=0;j<c;j++)

{

t[j][i]=a[i][j];

}

}

}

printf("\nTranspose of the matrix : \n"); for(i=0;i<c;i++)

{

for(j=0;j<r;j++)

{

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

}

printf("\n");

}

}

int determinant(int a[20][20],int row) //determinant

{

int det=0; if(row==2)

{

det=a[0][0]\*a[1][1] - a[1][0]\*a[0][1];

printf("\nDeterminant of the matrix : %d",det);

}

else if(row==3)

{

det=a[0][0]\*((a[1][1]\*a[2][2])-(a[2][1]\*a[1][2])) - a[0][1]\*(a[1][0]\*a[2][2]-

a[2][0]\*a[1][2]) + a[0][2]\*(a[1][0]\*a[2][1]-a[2][0] \* a[1][1]);

printf("\nDeterminant of the matrix : %d",det);

}

else

{

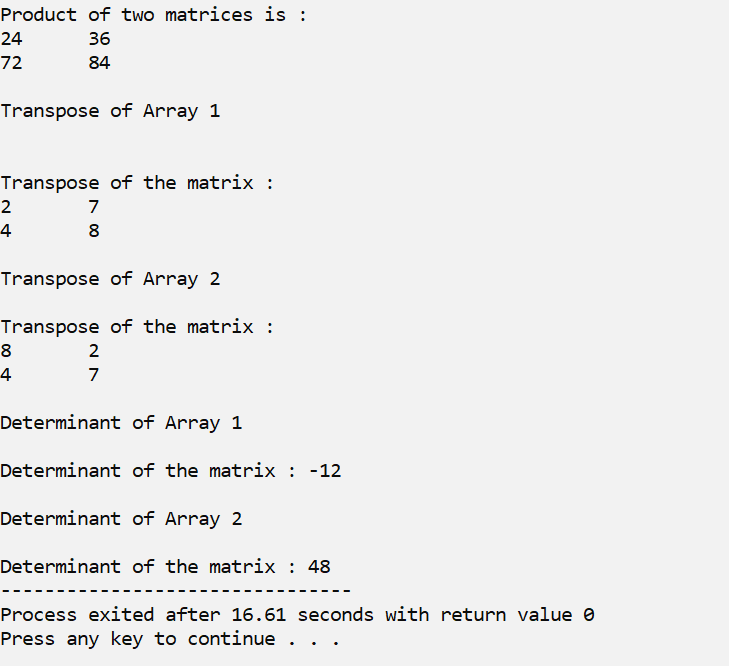
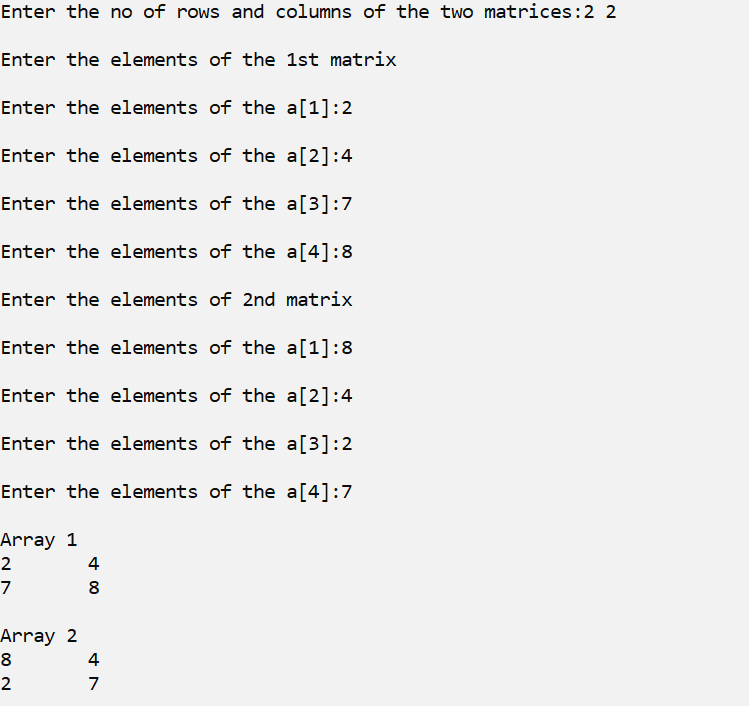
printf("\nDeterminant is not possible");

}

return 0;

}

**Output:**



|  |
| --- |
| **Program#8 Date:17/08/2022** |
| Allocate two dimensional array using pointers |

**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]);

## }

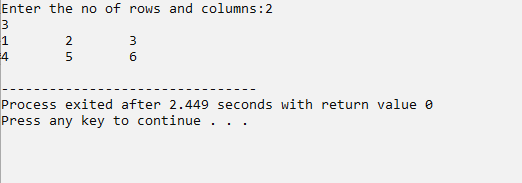
## printf("\n");

## }

## return 0;

## }

**OUTPUT**

****

|  |
| --- |
| **Program#9 Date:17/08/2022** |
| Display the array elements in the same order using a recursive function |

**Source Code:**

#include<stdio.h> int main()

{

int array[20],n,i;

printf("Enter the size of the array:"); scanf("%d",&n);

array[n]; for(i=0;i<n;i++)

{

printf("Enter array [%d] element: ",i+1); scanf("%d",&array[i]);

}

int start=0;

printf("\nThe Displayed elements in the array using Recursion function:\n"); recursive(array,n,start);

return 0;

}

int recursive(int array[10],int n,int start)

{

if(start<n)

{

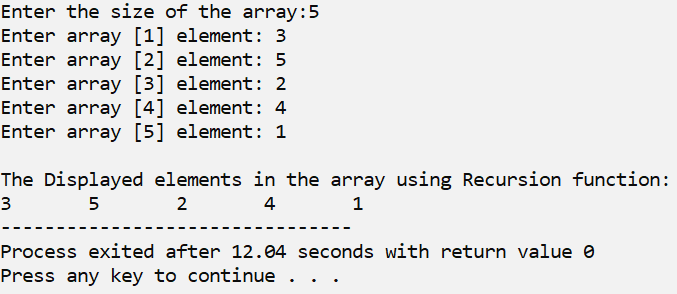
printf("%d\t",array[start]); start=start+1;

recursive(array,n,start);

}

}

**Output:**



|  |
| --- |
| **Program#10 Date:17/08/2022** |
| Display array elements in the reverse order using a recursive function |

**Source Code:**

#include<stdio.h> int main()

{

int array[20],n,i;

printf("Enter the size of the array:"); scanf("%d",&n);

array[n]; for(i=0;i<n;i++)

{

printf("Enter array [%d] element: ",i+1); scanf("%d",&array[i]);

}

printf("\nThe Elements in the Reverse Order:"); reverse(array,n);

}

int reverse(int array[],int n)

{

if(n>0)

{

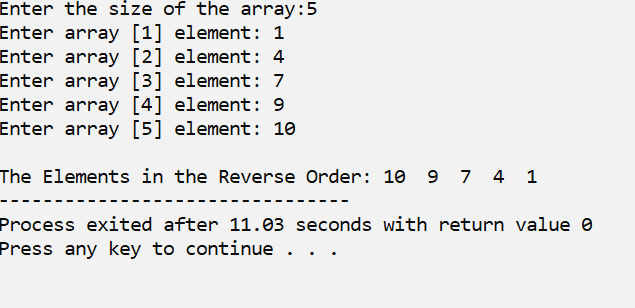
n=n-1;

printf(" %d ",array[n]); reverse(array,n);

}

}

**Output:**



**a.Just print it in the reverse order**

|  |
| --- |
| **Program#11 Date:23/08/2022** |
| Read a String and display it in the reverse order using   * 1. Just print it in the reverse order   2. Reverse the string in the same array itself |

**Source Code:**

#include<stdio.h> #include<stdlib.h> #include<string.h> int main()

{

char string[20]; int i,len=0;

printf("Enter the string :"); scanf("%s",&string); for(i=0;string[i]!='\0';i++)

{

len++;

}

printf("The string in reverse order :"); for(i=len-1;i>=0;i--)

{

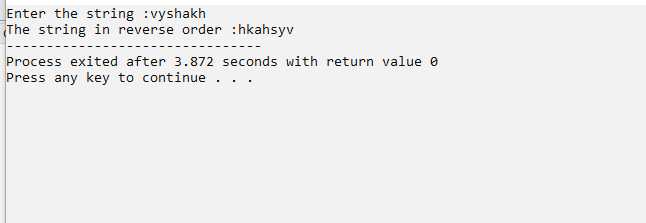
printf("%c",string[i]);

}

return 0;

}

**Output:**

****

**b.Reverse the string in the same array itself**

**Source Code:**

#include<stdio.h> #include<string.h> int main()

{

char string[50],temp[50]; int i,len=0;

printf("Enter the string:"); scanf("%s",&string); for(i=0;string[i]!='\0';i++)

{

len++;

}

printf("\nThe String in the reverse order is:"); for(i=0;i<len/2;i++)

{

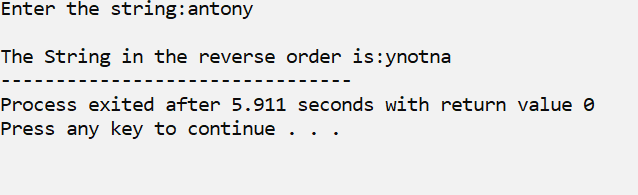
temp[i]=string[i]; string[i]=string[len-1-i]; string[len-1-i]=temp[i];

}

printf("%s",string); return 0;

}

**Output:**



|  |
| --- |
| **Program#12 Date:23/08/2022** |
| 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++)

{

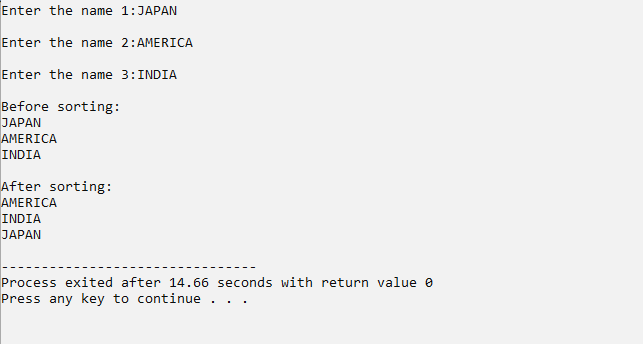
printf("%s\n",string[i]);

}

return 0;

}

**Output:**

****

|  |
| --- |
| **Program#13 Date:24/08/2022** |
| 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>

#include<conio.h>

struct date

{

int day;

int month;

int year;

};

void main()

{

struct date d1,d2;

printf("enter the first date in dd/mm/yyyy format\n");

scanf("%d%d%d",&d1.day,&d1.month,&d1.year);

printf("enter the second date in dd/mm/yyyy format\n");

scanf("%d%d%d",&d2.day,&d2.month,&d2.year);

if((d1.day==d2.day)&&(d1.month==d2.month)&&(d1.year==d2.year))

{

printf("EQUAL");

}

else

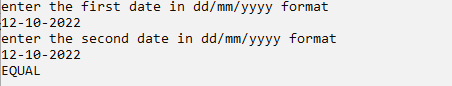
{

printf("NOT EQUAL");

}

}

**Output:**

****

|  |
| --- |
| **Program#14 Date:24/08/2022** |
| Define a structure for employees having eno,ename, esal and dno. Read n employees information and provide function for the following   * 1. Searching an employee by no   2. Sorting the employee by      1. Name      2. Salary   3. Deleting an employee |

**Sourcecode:**

#include<stdio.h>

#include<string.h>

struct employee

{

int eno;

char ename[25];

float esal;

int dno;

};

int n,i;

void main()

{

printf("enter the number of employees:");

scanf("%d",&n);

struct employee e[n],temp;

printf("enter the employee details:\n\n");

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

{

printf("enter employee no:");

scanf("%d",&e[i].eno);

printf("enter employee name:");

scanf("%s",&e[i].ename);

printf("enter employee salary:");

scanf("%f",&e[i].esal);

printf("enter employee dno:");

scanf("%d",&e[i].dno);

}

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

{

printf("\nemployee no:%d",e[i].eno);

printf("\nemployee name:%s",e[i].ename);

printf("\nemployee salary:%f",e[i].esal);

printf("\nemployee dno:%d\n",e[i].dno);

}

int key,flag=0;

printf("\nenter the employee no to be searched:");

scanf("%d",&key);

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

{

if(e[i].eno==key)

{

printf("\nemployee found:\n");

printf("\nemployee no:%d",e[i].eno);

printf("\nemployee name:%s",e[i].ename);

printf("\nemployee salary:%f",e[i].esal);

printf("\nemployee dno:%d",e[i].dno);

flag=1;

break;

}

}

if(flag==0)

printf("employee not found:\n");

int j;

printf("\n\nSorting data by name:\n");

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;

}

}

}

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

{

printf("\nemployee no:%d",e[i].eno);

printf("\nemployee name:%s",e[i].ename);

printf("\nemployee salary:%f",e[i].esal);

printf("\nemployee dno:%d\n",e[i].dno);

}

printf("\n\nSorting data by salary:\n");

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;

}

}

}

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

{

printf("\nemployee no:%d",e[i].eno);

printf("\nemployee name:%s",e[i].ename);

printf("\nemployee salary:%f",e[i].esal);

printf("\nemployee dno:%d\n",e[i].dno);

}

int del;

printf("\n\nEnter the employee number to be deleted:");

scanf("%d",&del);

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

{

if(e[i].eno==del)

{

n=n-1;

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

e[i]=e[i+1];

}

}

printf("\n\nData after deletion\n");

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

{

printf("\nemployee no:%d",e[i].eno);

printf("\nemployee name:%s",e[i].ename);

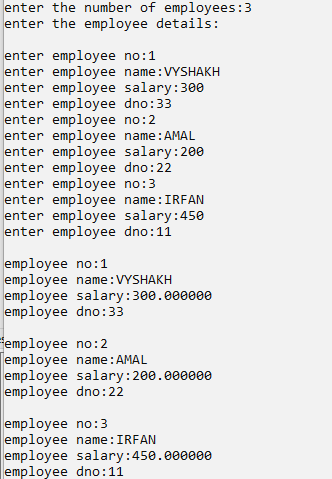
printf("\nemployee salary:%f",e[i].esal);

printf("\nemployee dno:%d\n",e[i].dno);

}

}

**Output**

****

|  |
| --- |
| **Program#15 Date:26/08/2022** |
| Implement a) malloc , b) calloc and c) free functions |

**Source Code:**

#include<stdio.h>

#include<stdlib.h>

int main()

{

int \*p,n,i,\*a;

printf("MALLOC\n");

printf("\*\*\*\*\*\*\*\*\*\*\n");

printf("enter the size:\n");

scanf("%d",&n);

p=(int \*)malloc(sizeof (int)\*n);

printf("memory allocated\n");

printf("Enter the elements:\n");

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

{

printf("Enter p[%d]:",i);

scanf("%d",p+i);

}

printf("The numbers entered are:");

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

{

printf("%d\t",\*(p+i));

}

free(p);

printf("\nCALLOC\n");

printf("\*\*\*\*\*\*\*\*\*\*\n");

printf("enter the size:\n");

scanf("%d",&n);

a = (int\*)calloc(n, sizeof(int));

printf("memory allocated\n");

printf("Enter the numbers:\n");

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

{

printf("Enter p[%d]:",i);

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

}

printf("The numbers entered are:");

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

{

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

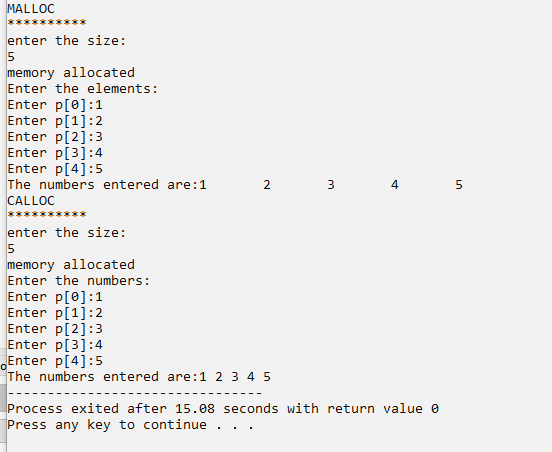
}

free(a);

return(0);

}

**OUTPUT**

****

|  |
| --- |
| **Program#16 Date:26/08/2022** |
| Use malloc to read n integers and find the mean. |

**Source Code:**

#include<stdio.h>

#include<stdlib.h>

int main()

{

int \*p,n,i,s=0;

float mean=0;

printf("Number of elements to be entered:");

scanf("%d",&n);

p=(int \*)malloc(sizeof(int)\*n);

printf("Enter %d numbers:\n",n);

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

{

printf("Enter p[%d]:",i);

scanf("%d",p+i);

}

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

{

s=s+\*(p+i);

}

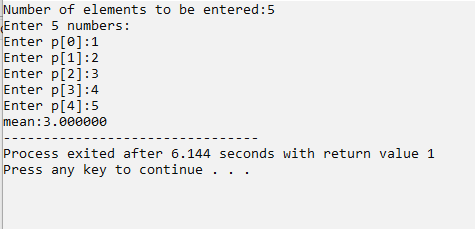
mean=s/n;

printf("mean:%f",mean);

free(p);

}

**OUTPUT**

****

|  |
| --- |
| **Program#17 Date:26/08/2022** |
| Use calloc to read n numbers and find the mode. |

**Source Code:**

#include<stdio.h>

#include<stdlib.h>

int main()

{

int \*p,n,i,j,count=0,mode=0,max=0,value=0;;

printf("Enter the size:");

scanf("%d",&n);

p=(int \*)calloc(n,sizeof (int));

printf("Enter %d numbers:\n",n);

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

{

printf("Enter p[%d]:",i);

scanf("%d",p+i);

}

printf("Elemnts are\n");

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

{

printf("p[%d] is %d\n",i,p[i]);

}

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

{

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

{

if(p[i]==p[j])

{

count=count+1;

}

}

if(count > max)

{

max=count;

value=p[i];

}

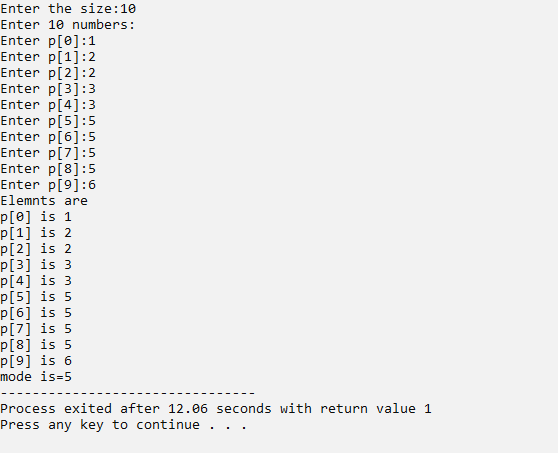
}

printf("mode is=%d",value);

free(p);

}

**Output:**

****

|  |
| --- |
| **Program#18 Date:26/08/2022** |
| 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<stdlib.h>

struct book

{

char book\_name[20];

char author\_name[20];

};

void read(struct book \*b1,int n)

{

int i;

printf("Enter the details of %d books",n);

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

{

printf("\nEnter book name:");

scanf("%s",&(b1+i)->book\_name);

printf("Enter author\_name:");

scanf("%s",&(b1+i)->author\_name);

}

}

void display(struct book \*b1,int n)

{

int i;

printf("\n\*\*\*\*Boook details\*\*\*\*\n");

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

{

printf("Book name:%s\n",(b1+i)->book\_name);

printf("Book author name:%s\n",(b1+i)->author\_name);

}

}

int main()

{

struct book \*b1;

int n;

printf("Enter number of books:");

scanf("%d",&n);

b1 = (struct book \*)calloc(n, sizeof(struct book));

read(b1,n);

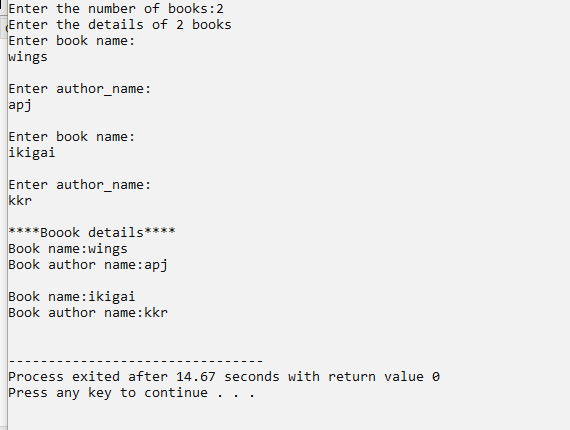
display(b1,n);

free(b1);

return 0;

}

**OUTPUT**

****

|  |
| --- |
| **Program#19 Date:26/08/2022** |
| 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++)

{

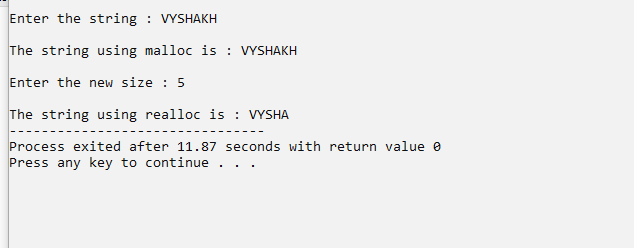
printf("%c",\*(ptr+i));

}

free(ptr); return 0;

}

**OUTPUT**



|  |
| --- |
| **Program#20 Date:14/09/2022** |
| Implement stack operations using array |

**Source Code**

#include<stdio.h>

#define N 5

int stack[N];

int top=-1;

void push(int e)

{

if(top+1==N)

{

printf("Stack is full");

}

else

{

top=top+1;

stack[top]=e;

printf("Element is pushed\n");

}

}

void pop()

{

if(top==-1)

{

printf("Stack is empty");

}

else

{

printf("\npopped %d",stack[top]);

top=top-1;

}

}

void peak()

{

if(top==-1)

{

printf("stack is empty");

}

else

{

printf("Top Element is %d\n",stack[top]);

}

}

void main()

{

int ch,e;

while(ch!=4)

{

printf("\n1.push\n2.pop\n3.peak\n4.exit\nEnter you choice:");

scanf("%d",&ch);

switch(ch)

{

case 1: printf("Enter the element:\n");

scanf("%d",&e);

push(e);

break;

case 2: pop();

break;

case 3: peak();

break;

case 4: break;

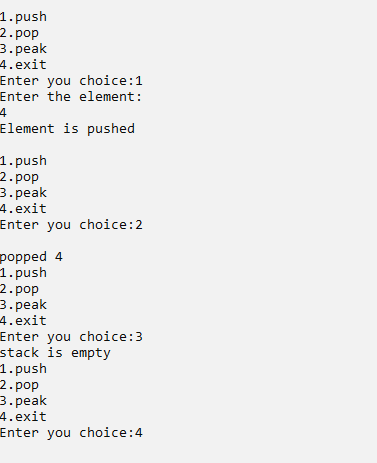
default:printf("wrong choice");

}

}

}

**Output:**

****

|  |
| --- |
| **Program#21 Date:14/09/2022** |
| Reverse a string using Stack |

**Source Code:**

#include <stdio.h>

#include <string.h>

#define max 100

int top,stack[max];

void push(char x){

// Push(Inserting Element in stack) operation

if(top == max-1){

printf("stack overflow");

} else {

stack[++top]=x;

}

}

void pop(){

// Pop (Removing element from stack)

printf("%c",stack[top--]);

}

void main()

{

char str[100];

printf(" enter the string ");

scanf("%s",&str);

int len=strlen(str);

int i;

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

{

push(str[i]);

}

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

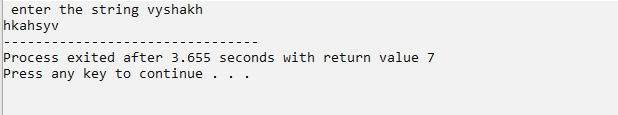
{

pop(str[i]);

}

}

**OUTPUT**

****

|  |
| --- |
| **Program#22 Date:17/09/2022** |
| Convert an expression from infix to postfix using stack |

**Source Code:**

#include<stdio.h>

#include<ctype.h>

char stack[100];

int top = -1;

void push(char x)

{

stack[++top] = x;

}

char pop()

{

if(top == -1)

return -1;

else

return stack[top--];

}

int priority(char x)

{

if(x == '(')

return 0;

if(x == '+' || x == '-')

return 1;

if(x == '\*' || x == '/')

return 2;

return 0;

}

int main()

{

char exp[100];

char \*e, x;

printf("Enter the expression : ");

scanf("%s",exp);

printf("\n");’

e = exp;

while(\*e != '\0')

{

if(isalnum(\*e))

printf("%c ",\*e);

else if(\*e == '(')

push(\*e);

else if(\*e == ')')

{

while((x = pop()) != '(')

printf("%c ", x);

}

else

{

while(priority(stack[top]) >= priority(\*e))

printf("%c ",pop());

push(\*e);

}

e++;

}

while(top != -1)

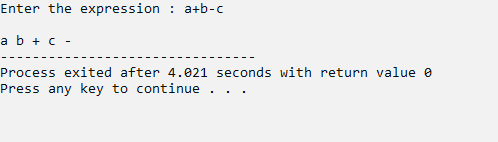
{

printf("%c ",pop());

}return 0;

}

**Output:**

****

|  |
| --- |
| **Program#23 Date:17/09/2022** |
| Evaluate an expression using stack |

**Source Code:**

#include<stdio.h>

int stack[20];

int top = -1;

void push(int x)

{

stack[++top] = x;

}

int pop()

{

return stack[top--];

}

int main()

{

char exp[20];

char \*e;

int n1,n2,n3,num;

printf("Enter the expression :: ");

scanf("%s",exp);

e = exp;

while(\*e != '\0')

{

if(isdigit(\*e))

{

num = \*e - 48;

push(num);

}

else

{

n1 = pop();

n2 = pop();

switch(\*e)

{

case '+':

{

n3 = n1 + n2;

break;

}

case '-':

{

n3 = n2 - n1;

break;

}

case '\*':

{

n3 = n1 \* n2;

break;

}

case '/':

{

n3 = n2 / n1;

break;

}

}

push(n3);

}

e++;

}

printf("\nThe result of expression %s = %d\n\n",exp,pop());

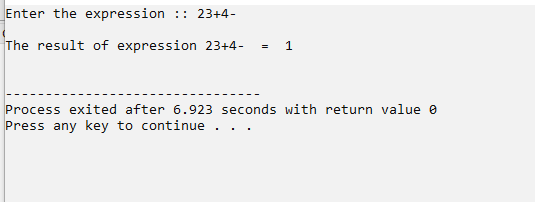
return 0;

}

}

}

**OUTPUT**

****

|  |
| --- |
| **Program#24 Date:17/09/2022** |
| 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]);

}

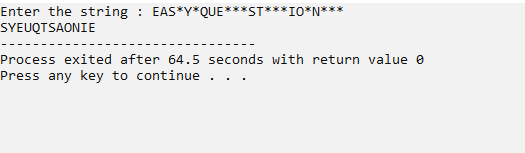
i++;

}

return 0;

}

**OUTPUT**

****

|  |
| --- |
| **Program#25 Date:20/09/2022** |
| 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++)

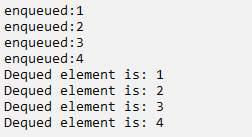
{

dequeue();

}

}

**OUTPUT**

****

|  |
| --- |
| **Program#26 Date:20/09/2022** |
| 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()

{

int i=0;

for(i=0;i<6;i++)

{

enqueue(i);

}

for(i=0;i<6;i++)

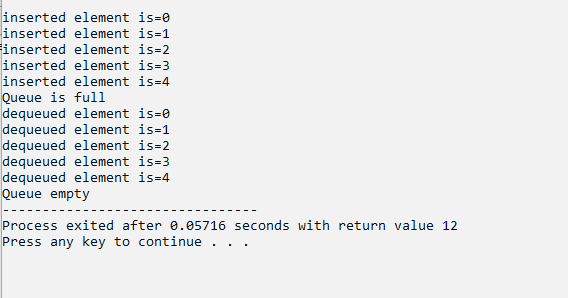
{

dequeue();

}

}

**OUTPUT**

****

|  |
| --- |
| **Program#27 Date:20/09/2022** |
| 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 YOUR CHOICE\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

## 

|  |
| --- |
| **Program#28 Date:29/09/2022** |
| Demonstrate 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);

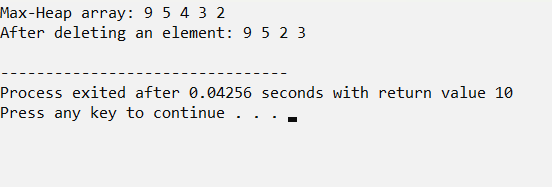
deleteRoot(array, 4);

printf("After deleting an element: ");

printArray(array, size);

}

**OUTPUT**



|  |
| --- |
| **Program#29 Date:29/09/2022** |
| Implement heap sort. |

**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)

{

for (int i = n / 2 - 1; i >= 0; i--) heapify(arr, n, i);

for (int i = n - 1; i >= 0; i--)

{

swap(&arr[0], &arr[i]); heapify(arr, i, 0);

}

}

void printArray(int arr[], int n)

{

for (int i = 0; i < n; ++i) printf("%d ", arr[i]); printf("\n");

}

int main()

{

int arr[SIZE];

printf("\nArray Size : %d\n",SIZE); for(int i=0; i<SIZE; i++)

{

printf("\nEnter the array [%d] element --> ",i+1); scanf("%d",&arr[i]);

}

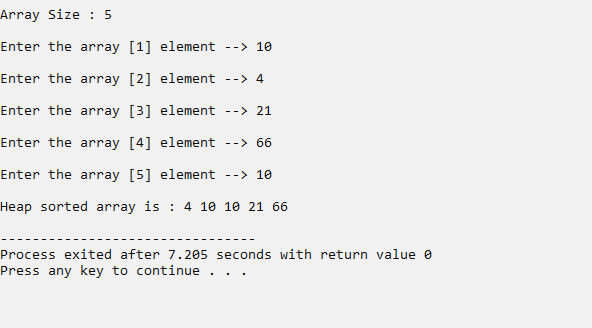
int n = sizeof(arr) / sizeof(arr[0]); heapSort(arr, n);

printf("\nHeap sorted array is : "); printArray(arr, n);

return 0;

}

**OUTPUT**



|  |
| --- |
| **Program#30 Date:06/10/2022** |
| 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;

}

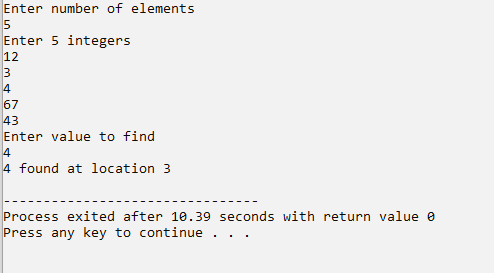
if(low > high)

printf("Not found! %d isn't present in the list\n", key);

return 0;

}

**OUTPUT**

****

|  |
| --- |
| **Program#31 Date:06/10/2022** |
| Implement various sorting Algorithms   * 1. Bubble Sort   2. Selection Sort   3. 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:

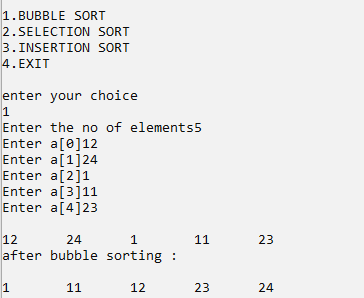
printf("wrong choice");

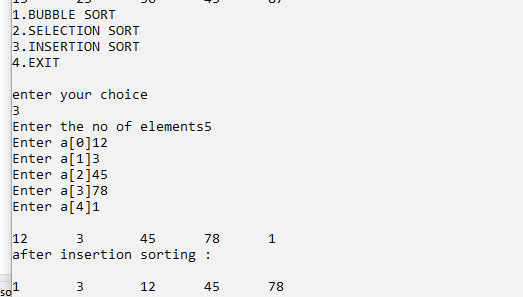
}

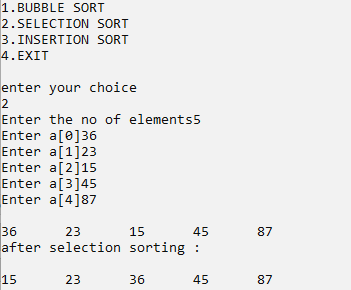
}

}

**OUTPUT**







|  |
| --- |
| **Program#32 Date:11/10/2022** |
| 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]);

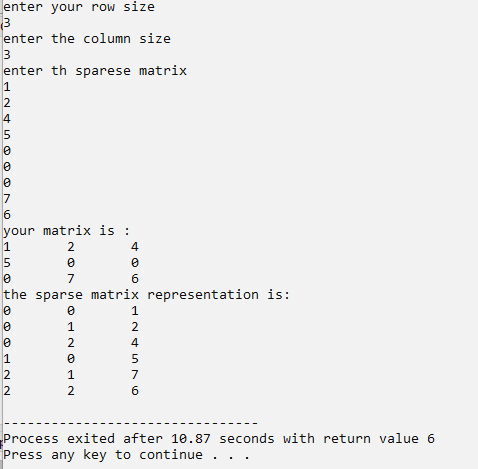
}

printf("\n");

}

}

**output**

****

|  |
| --- |
| **Program#33 Date:14/10/2022** |
| Demonstrate Circular Linked List |

**Source Code:**

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

struct node

{

int data;

struct node \*next;

};

typedef struct node clist; clist \*head=NULL;

void insertion()

{

clist \*t; if(head==NULL)

{

}

else

{

head=(clist\*)malloc(sizeof(clist)); printf("\nEnter the element to be inserted : "); scanf("%d",&head->data);

head->next=head;

t=head;

while(t->next!=head)

{

t=t->next;

}

t->next=(clist\*)malloc(sizeof(clist)); printf("\nEnter the element to be inserted : "); scanf("%d",&t->next->data);

t->next->next=head;

}

}

void deletion()

{

int ele; clist \*t;

printf("\nEnter the element to be deleted : "); scanf("%d",&ele);

if(head==NULL)

printf("\nCircular Linked List is Empty"); else if(head->data==ele)

{

if(head->next==head) head=NULL;

else

{

}

}

else

{

for(t=head; t->next!=head; t=t->next); t->next=head->next;

head=head->next;

for(t=head; t->next!=head && t->next->data!=ele; t=t->next); if(t->next==head)

printf("\nElement not found"); else

{

t->next=t->next->next;

}

}

}

void display()

{

printf("\n"); clist \*t; t=head; if(t==NULL)

printf("\nCircular List is Empty"); else

{

do

{

printf("%d ",t->data); t=t->next;

}while(t!=head);

}

}

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 : insertion(); break;

case 2 : deletion(); break;

case 3 : display(); break;

case 4 : break;

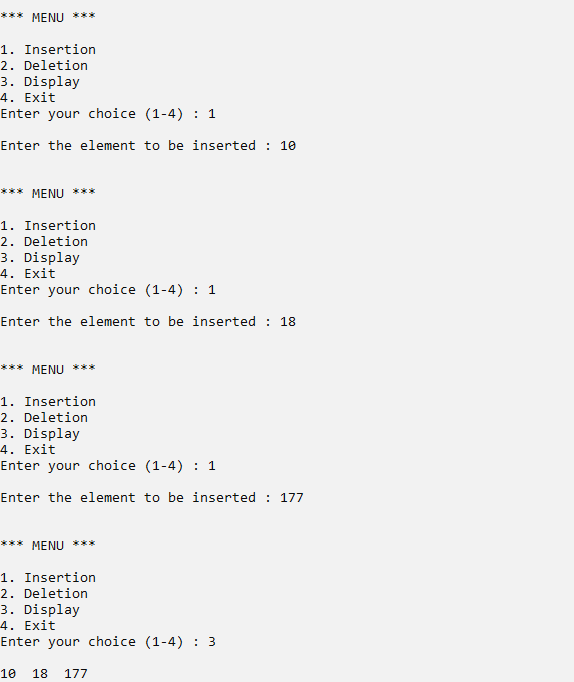
default : printf("\nInvalid Choice\n");

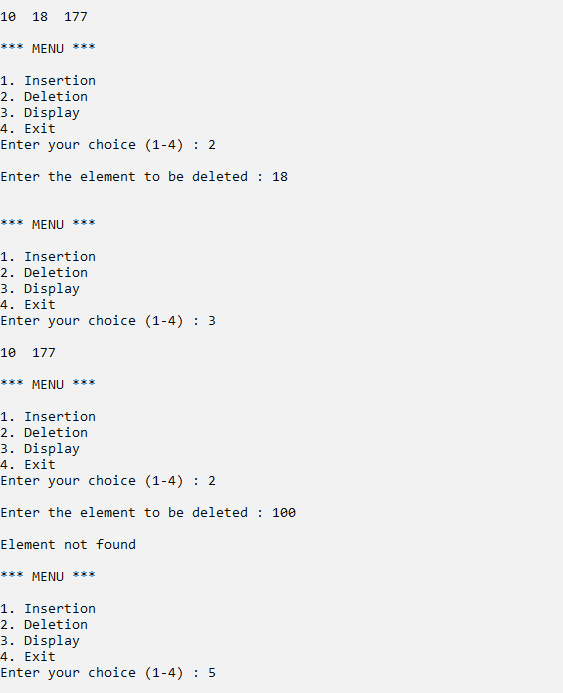
}

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

}

**Output:**





|  |
| --- |
| **Program#34 Date:14/10/2022** |
| Demonstrate Doubly Linked List |

**Source Code:**

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

struct node

{

int data;

struct node \*next,\*prev;

};

typedef struct node dll; dll \*head=NULL;

void insertion()

{

dll \*t; if(head==NULL)

{

}

else

{

}

}

head=(dll\*)malloc(sizeof(dll)); printf("\nEnter the element to be inserted : "); scanf("%d",&head->data);

head->next=NULL; head->prev=NULL;

for(t=head; t->next!=NULL; t=t->next); t->next=(dll\*)malloc(sizeof(dll));

printf("\nEnter the element to be inserted : "); scanf("%d",&t->next->data);

t->next->next=NULL; t->next->prev=t;

void deletion()

{

int ele; dll \*t;

printf("\nEnter the element to be deleted : "); scanf("%d",&ele);

if(head==NULL)

printf("\nDoubly Linked List is Empty"); else if(head->data==ele)

{

if(head->next==NULL) head=NULL;

}

else

{

else

{

}

head=head->next; head->prev=NULL;

for(t=head; t!=NULL && t->data!=ele; t=t->next); if(t==NULL)

printf("\nElement not found"); else if(t->next==NULL)

t->prev->next=NULL; else

{

t->next->prev=t->prev; t->prev->next=t->next;

}

}

}

void display()

{

printf("\n"); dll \*t;

for(t=head; t!=NULL; t=t->next)

{

printf("%d ",t->data);

}

}

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 : insertion(); break;

case 2 : deletion();

break;

case 3 : display(); break;

case 4 : break;

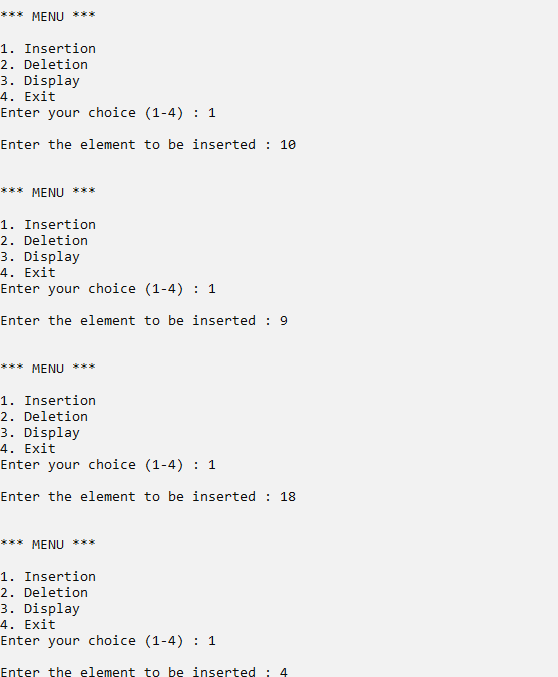
default : printf("\nInvalid Choice\n");

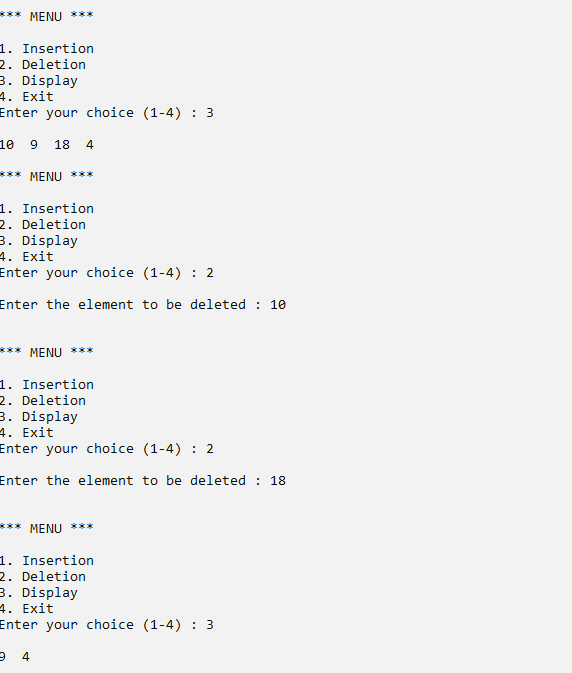
}

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

}

**Output:**





|  |
| --- |
| **Program#35 Date:14/10/2022** |
| 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**

****

|  |
| --- |
| **Program#36 Date:14/10/2022** |
| 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;

}

}

/\*If second list has finished and elements left in first list\*/

while(p1!=NULL)

{

start3=insert(start3,p1->info);

p1=p1->link;

}

/\*If first list has finished and elements left in second list\*/

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;

}/\*End of create\_slist()\*/

struct node \*insert\_s(struct node \*start,int data)

{

struct node \*p,\*tmp;

tmp=(struct node \*)malloc(sizeof(struct node));

tmp->info=data;

/\*list empty or data to be added in beginning \*/

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);

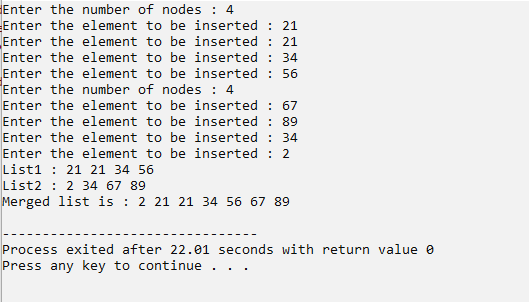
p=p->link;

}

printf("\n");

}

**Output:**



|  |
| --- |
| **Program#37 Date:19/10/2022** |
| Linked Stack |

**Source Code:**

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

struct stack

{

int data;

struct stack \*next;

};

typedef struct stack stack; stack \*top=NULL;

void push()

{

stack \*t; t=(stack\*)malloc(sizeof(stack));

printf("\nEnter the element to be inserted : "); scanf("%d",&t->data);

t->next=top; top=t;

}

void pop()

{

if(top==NULL) printf("\nEmpty Stack\n"); else

{

printf("%d is deleted\n",top->data); top=top->next;

}

}

void peep()

{

if(top==NULL) printf("\nEmpty Stack\n"); else

printf("\nThe top element is : %d\n",top->data);

}

int main()

{

int ch; do

{

printf("\n\*\*\* MENU \*\*\*\n"); printf("\n1. Push (Insertion)"); printf("\n2. Pop (Deletion)");

printf("\n3. Peep (Displaying the top element)"); printf("\n4. Exit");

printf("\nEnter your choice (1-4) : "); scanf("%d",&ch);

switch(ch)

{

case 1 : push();

break; case 2 : pop();

break; case 3 : peep();

break; case 4 : break;

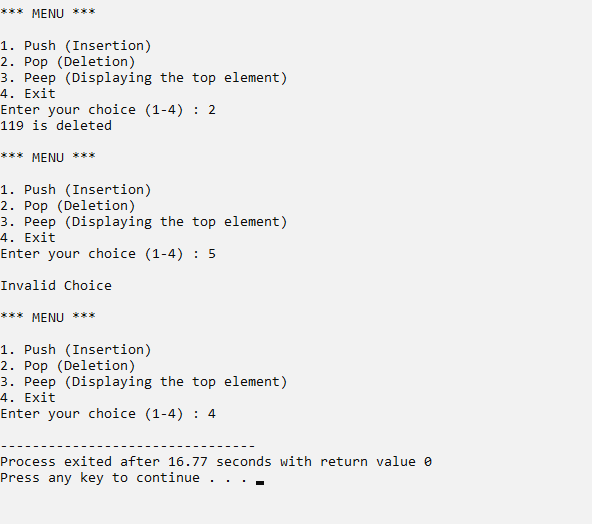
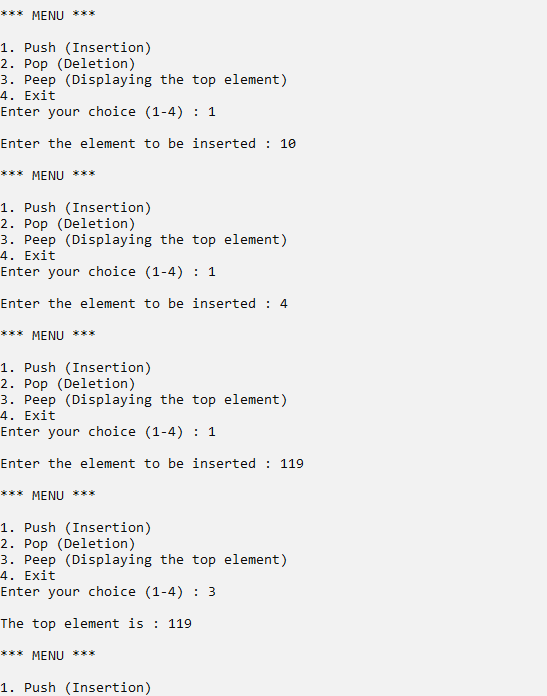
default : printf("\nInvalid Choice\n");

}

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

}

**Output:**



|  |
| --- |
| **Program#38 Date:19/10/2022** |
| 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: ");

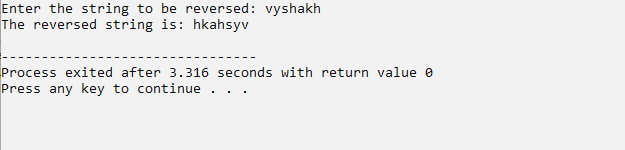
scanf("%s", string);

rev(string);

return 0;

}

**Output:**



|  |
| --- |
| **Program#39 Date:19/10/2022** |
| Linked Queue |

**Source Code:**

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

struct queue

{

int data;

struct queue \*next;

};

typedef struct queue queue; queue \*front=NULL; queue \*rear=NULL;

void enqueue()

{

if(front==NULL)

{

}

else

{

}

}

front=(queue\*)malloc(sizeof(queue)); printf("\nEnter the element to be inserted : "); scanf("%d",&front->data);

front->next=NULL; rear=front;

rear->next=(queue\*)malloc(sizeof(queue)); printf("\nEnter the element to be inserted : "); scanf("%d",&rear->next->data);

rear->next->next=NULL; rear=rear->next;

void dequeue()

{

if(front==NULL) printf("\nEmpty Queue\n"); else if(front==rear)

{

}

else

{

printf("\n%d is deleted\n",front->data); free(front);

front=rear=NULL;

printf("%d is deleted\n",front->data); queue \*t=front;

front=front->next; free(t);

}

}

void display()

{

printf("\n"); queue \*t=front; while(t!=NULL)

{

printf("%d ",t->data); t=t->next;

}

}

int main()

{

int ch; do

{

printf("\n");

printf("\n\*\*\* MENU \*\*\*\n"); printf("\n1. Enqueue (Insertion)"); printf("\n2. Dequeue (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;

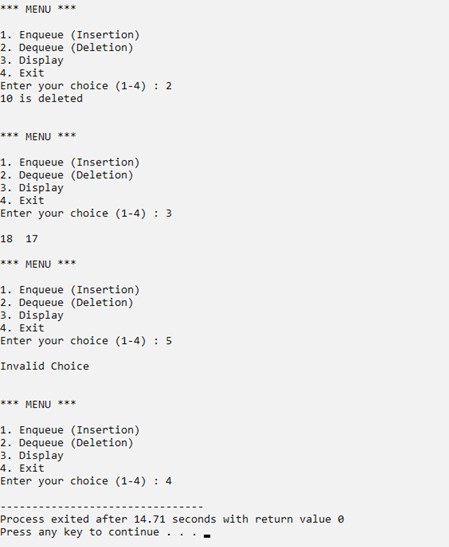
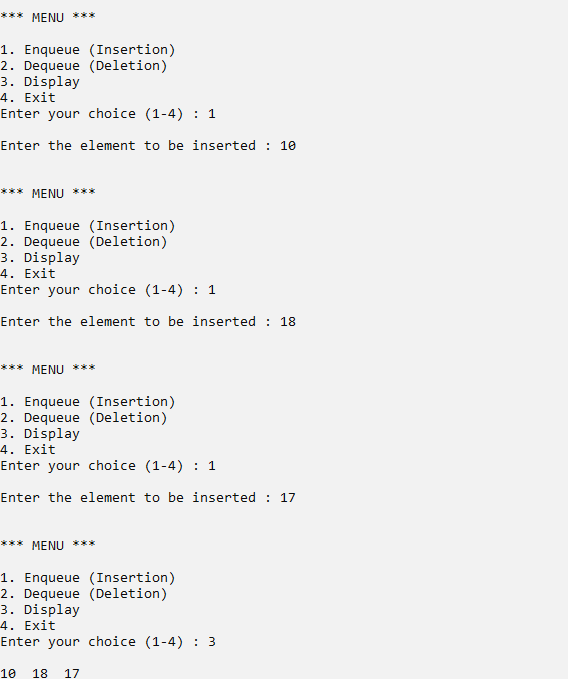
default : printf("\nInvalid Choice\n");

}

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

}

**Output:**



|  |
| --- |
| **Program#40 Date:19/10/2022** |
| 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)

{

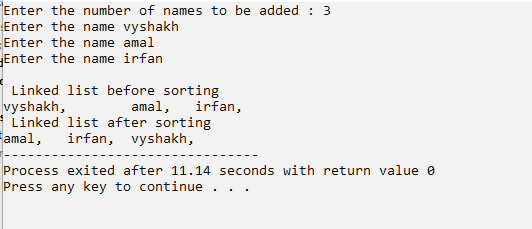
printf("%s,\t", temp->data);

temp = temp->next;

}

}

**Output:**



|  |
| --- |
| **Program#41 Date:19/10/2022** |
| 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

{

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;

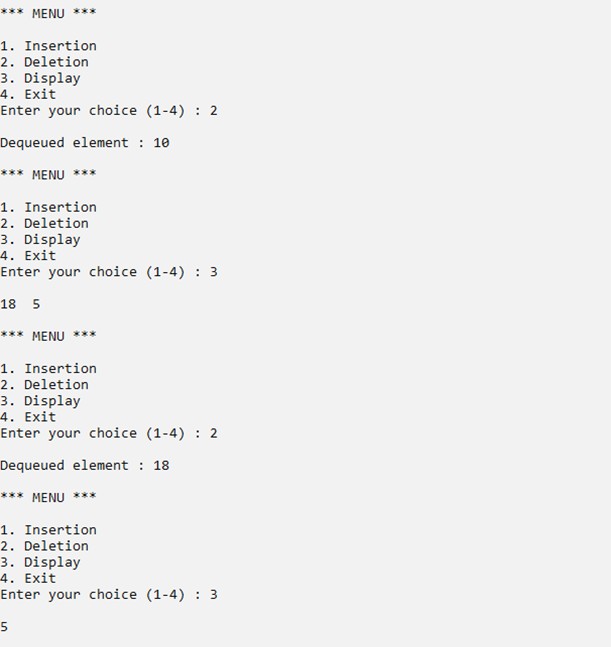
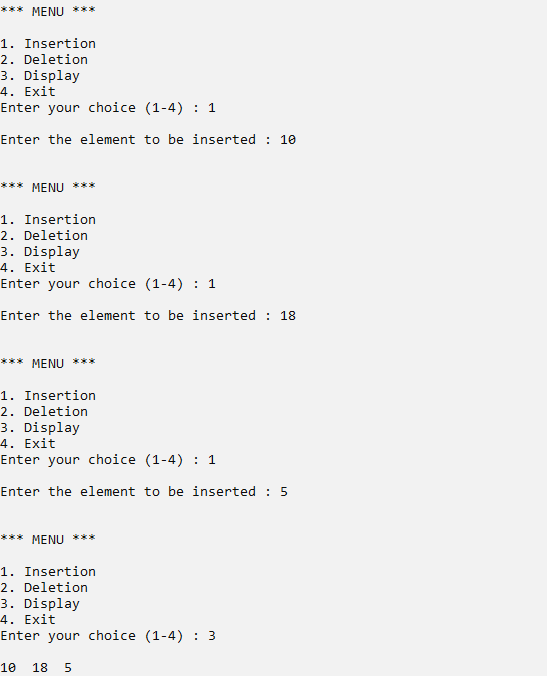
default : printf("\nInvalid Choice\n");

}

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

}

**Output:**



|  |
| --- |
| **Program#42 Date:21/10/2022** |
| 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");

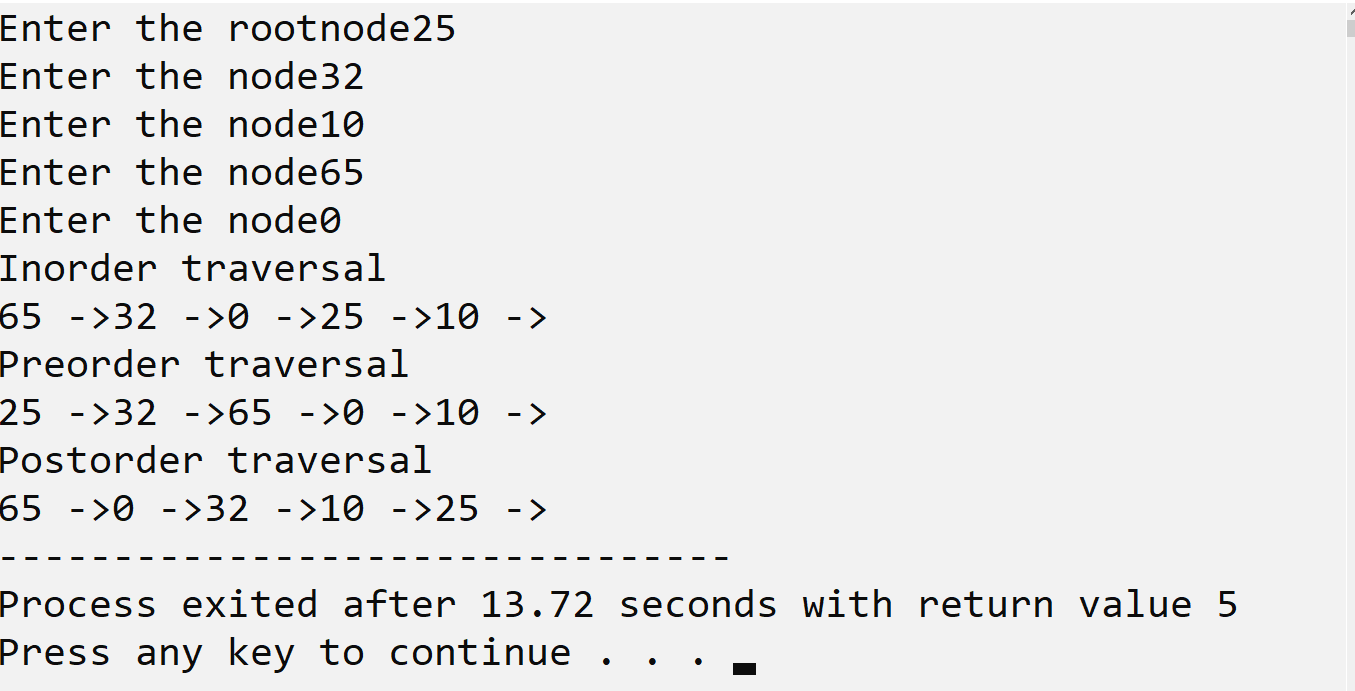
preorderTraversal(root);

printf("\nPostorder traversal \n");

postorderTraversal(root);

}

**Output:**



**Program #33**

|  |
| --- |
| **Program#43 Date:21/10/2022** |
| 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();

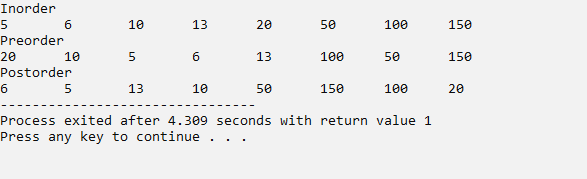
free(head);

free(top);

free(tree);

}

**OUTPUT**

****

|  |
| --- |
| **Program#44 Date:21/10/2022** |
| 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("vyshakh");

insert("amal");

insert("rajagiri");

insert("irfan");

insert("sreekuttan");

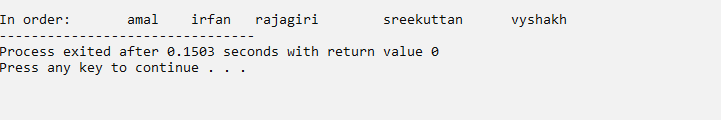
printf("\nIn order: ");

inorder(root);

return 0;

}

**Output:**



|  |
| --- |
| **Program#45 Date:21/10/2022** |
| 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);

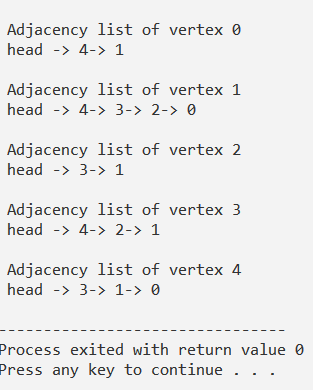
printGraph(graph);

return 0;

}

**OUTPUT:**

**OUTPUT:**



|  |
| --- |
| **Program#46 Date:21/10/2022** |
| Personal road map (Project) |

**SOURCE CODE:**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

typedef struct linked\_list

{

char pname[10];

int km;

struct linked\_list\*next ;

}node;

node \*head;

void destination(int x);

int countnode();

void Insertion\_at\_Beggining();

int Insertion\_at\_middle(int c);

void Insertion\_at\_End();

void Display();

void Deletion\_at\_Beggining();

int Deletion\_at\_Middle(int c);

void Deletion\_at\_End();

int search();

node \*getnode()

{

node \*ptr;

char p[20];

int n;

ptr=(node\*)malloc(sizeof(node));

printf("\nEnter the Destination to be planned \t\t:");

scanf("%s",&p);

strcpy(ptr->pname,p);

printf("\nEnter the Kilometors to be traveled from the last destination :");

scanf("%d",&n);

ptr->km=n;

return ptr;

}

int main()

{

int ch,n,c;

printf("\n\n\n");

printf(" WELCOME TO ROADMAP APPLICATION \n");

printf("\n\n\n\n");

printf("Enter the Number of Destinations to be added : ");

scanf("%d",&n);

printf("\n\n\n");

destination(n);

Display();

c=countnode();

while(ch!=10)

{

printf("\n\nEnter the Other Operation to be done ");

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

printf("1.Insertion at Beggning \n2.Insertion at middle \n3.Insertion at End \n4.Deletion at Beggining \n5.Deletion at Middle \n6.Deletion at End \n7.Display The Destinations \n8.Search \n9.Count of Destinations \n10.Exit\n\n\n\n\n ");

printf("ENTER YOUR CHOICE : ");

scanf("%d",&ch);

switch (ch)

{

case 1:

{

Insertion\_at\_Beggining();

break;

}

case 2:

{

Insertion\_at\_middle(c);

break;

}

case 3:

{

Insertion\_at\_End();

break;

}

case 4:

{

Deletion\_at\_Beggining();

break;

}

case 5:

{

Deletion\_at\_Middle(c);

break;

}

case 6:

{

Deletion\_at\_End();

break;

}

case 7:

{

Display();

break;

}

case 8:

{

search();

break;

}

case 9:

{

countnode();

break;

}

case 10:

{

break;

}

default :

{

printf("Invalid Entry");

}

}

}

return 0;

}

int countnode()

{

int count=0;

node \*temp;

temp=head;

while (temp!=NULL)

{

count++;

temp=temp->next;

}

printf("\nCount of Destinations %d ",count);

return count;

}

void destination(int n)

{

int i;

node \*temp;

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

{

node \*newnode;

newnode=getnode();

if(head==NULL)

{

newnode->next=NULL;

head=newnode;

}

else

{

temp=head;

while(temp->next!=NULL)

temp=temp->next;

temp->next=newnode;

newnode->next=NULL;

}

printf("One Destionation Inserted \n");

}

}

void Display()

{

node \*temp;

if(head==NULL)

{

printf("\nNo Destinations ");

}

else

{

temp=head;

printf("\nThe Destinations are : ");

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

while(temp!=NULL)

{

printf("%s\t\t%d\n",temp->pname,temp->km);

temp=temp->next;

}

}

}

void Insertion\_at\_Beggining()

{

node \*newnode;

int c;

newnode=getnode();

if(head==NULL)

{

newnode->next=NULL;

head=newnode;

}

else

{

newnode->next=head;

head=newnode;

}

printf("Enter the new Distance in kilometer from %s to %s :",newnode->pname,newnode->next->pname);

scanf("%d",&c);

newnode->next->km=c;

printf("Destination Inserted");

}

void Insertion\_at\_End()

{

node \*newnode,\*temp;

newnode=getnode();

if(head==NULL)

{

newnode->next=NULL;

head=newnode;

}

else

{

temp=head;

while(temp->next!=NULL)

temp=temp->next;

temp->next=newnode;

newnode->next=NULL;

}

printf("\nOne Destionation Inserted \n");

}

int Insertion\_at\_middle(int c)

{

char p[20];

int a,i;

node \*temp,\*newnode;

printf("Enter the Destination after which you want to insert ");

scanf("%s",&p);

temp=head;

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

{

a=strcmp(p,temp->pname);

if(a==0)

{

newnode=getnode();

newnode->next=temp->next;

temp->next=newnode;

printf("Enter the new Distance in kilometer from %s to %s :",newnode->pname,newnode->next->pname);

scanf("%d",&c);

newnode->next->km=c;

return 0;

}

temp=temp->next;

}

}

void Deletion\_at\_Beggining()

{

node \*temp;

temp=head;

if(temp==NULL)

{

printf("\nNo Destination ");

}

else

{

temp=head;

head=temp->next;

free(temp);

printf("\nOne Destination Deleted from Front \n");

}

}

void Deletion\_at\_End()

{

node \*temp,\*pre;

if(head==NULL)

{

printf("\nNo Destination ");

}

else if(head->next == NULL)

{

head = NULL;

free(head);

printf("\nOnly destination Deleted ");

}

else

{

temp=head;

while(temp->next!=NULL)

{

pre=temp;

temp=temp->next;

}

pre->next=NULL;

free(temp);

printf("\nLast Destination Deleted ");

}

}

int Deletion\_at\_Middle(int c)

{

node\*temp,\*pre;

int i,x;

char a[20];

printf("Enter the Destination to be deleted ");

scanf("%s",&a);

temp=head;

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

{

if(temp==NULL)

{

printf("No Destinations");

return 0;

}

else if (temp->next==NULL)

{

head = NULL;

free(head);

printf("\nOnly destnation Deleted ");

return 0;

}

else

{

while(temp!=NULL)

{

x=strcmp(a,temp->pname);

if(x==0)

{

pre->next=temp->next;

free(temp);

printf("Destination %s Deleted ",a);

return 0;

}

pre=temp;

temp=temp->next;

}

}

}

}

int search()

{

int i,x;

char s[20];

node \*temp;

printf("Enter the Destination to be searched ");

scanf("%s",&s);

temp=head;

if(temp==NULL)

{

printf("\nNO DESTINATION ");

}

else

{

while (temp!=NULL)

{

x=strcmp(s,temp->pname);

if (x==0)

{

printf("Destination %s found %d kilometers Last Destination ",s,temp->km);

}

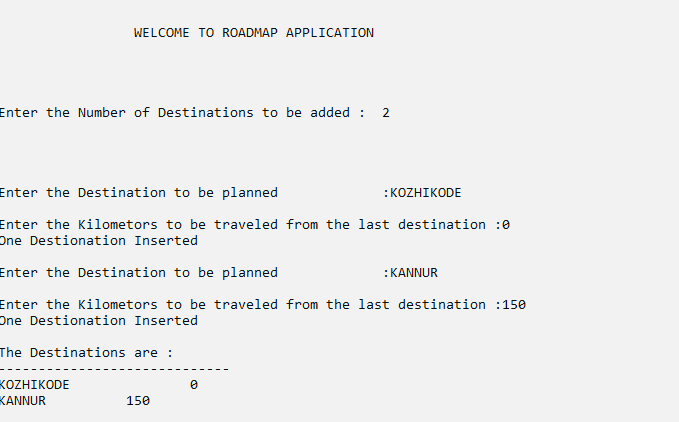
temp=temp->next;

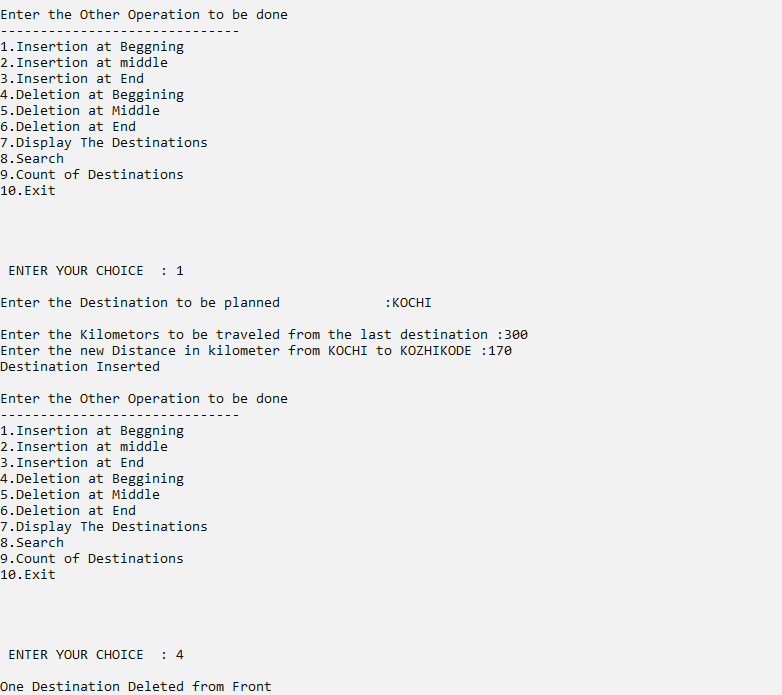
}

}

}

**OUTPUT:**

****

****