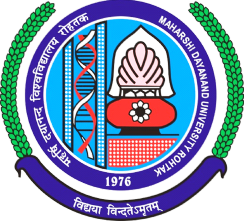
Maharshi Dayanand University



practical file of dsa lab using “c”

Submitted in partial fulfillment of the requirement in COMPUTER SCIENCE ENGINEERING (session 2019-2023)

**Name: Bazgha**

**Subject Code: LC-CSE-213G**

**Registration Number: 191380214**

|  |  |  |  |
| --- | --- | --- | --- |
| **INDEX** | | | |
| S.NO | DATE | EXPERIMENT | SIGN |
| 1 | 31/8/2020 | Write a program to insert and print the elements of a linear array. |  |
| 2 | 1/9/2020 | Write a program to delete an element of a linear array. By using  -position  -value |  |
| 3 | 2/9/2020  3/9/2020 | WAP to find the location of a given element in the array |  |
| 4 | 7/9/2020  8/9/2020 | WAP to insert an element into a linked list |  |
| 5 | 9/9/2020  9/9/2020 | WAP to delete an element from the linked list |  |
| 6 | 10/9/2020 | WAP to search and display element of a linked list |  |
| 7 | 14/9/2020  15/9/2020 | WAP to implement all the stack operations:  -push  -pop  -display |  |
| 8 | 16/9/2020  17/9/2020 | WAP to implement various operations on a linear queue using linear array. |  |
| 9 | 21/9/2020  22/9/2020 | WAP to implement various operations on a circular queue using linear array. |  |
| 10 | 23/9/2020  24/9/2020 | WAP to implement various operations on a queue using linear linked list. |  |
| 11 | 28/9/20  29/9/20 | WAP to demonstrate the use of linear search of a given element in an array. |  |
| 12 | 5/10/20  6/10/20 | WAP to demonstrate the use of binary search of a given element in an array. |  |
| 13 | 12,13,14,15/10/20 | WAP to illustrate the implementation of different operations on a BST |  |
| 14 | 21/10/20  22/10/20 | WAP to sort an array of integers in ascending order using bubble sort |  |
| 15 | 2/11/20  3/11/20 | WAP to sort an array of integers in ascending order using selection sort |  |
| 16 | 9/11/20  10/11/20 | WAP to sort an array of integers in ascending order using insertion sort |  |
| 17 | 1/12/20  7/12/20 | WAP to sort an array of integers in ascending order using merge sort |  |
| 18 | 14,15/12/20 | WAP to sort an array of integers in ascending order using quick sort |  |
| 19 | 22,23/12/20 | WAP to sort an array of integers in ascending order using heap sort |  |
| 20 | 29,30/12/20 | WAP to sort an array of integers in ascending order using shell sort |  |

**BAZGHA RAZI**

**Date Of Program**:31/08/2020 **Program Number**:01

**Roll Number**:

**AIM:** Write a program to insert and print the elements of a linear array.

**PROGRAM:**

#include<stdio.h>

int main()

{

int a[20],n,x,i,pos=0;

printf("Enter size of array:");

scanf("%d",&n);

printf("Enter the array in ascending order:\n");

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

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

printf("\nEnter element to insert:");

scanf("%d",&x);

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

if(a[i]<=x&&x<a[i+1])

{

pos=i+1;

break;

}

for(i=n+1;i>pos;--i)

a[i]=a[i-1];

a[pos]=x;

printf("\nArray after inserting element:");

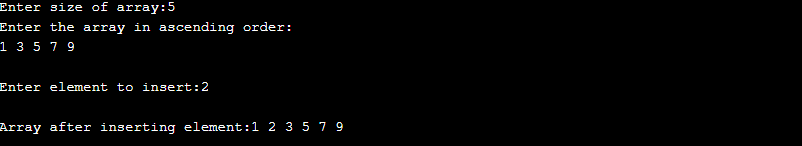
for(i=0;i<n+1;i++)

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

return 0;

}

**OUTPUT:**



**BAZGHA RAZI**

**Date Of Program**:01/09/2020 **Program Number**:02

**Roll Number**:

**AIM:** Write a program to delete an element of a linear array. By using

-position

-value

**PROGRAM:**

#include <stdio.h>

int main()

{

int array[100], position, c, n;

printf("Enter number of elements in array\n");

scanf("%d", &n);

printf("Enter %d elements\n", n);

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

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

printf("Enter the location where you wish to delete element\n");

scanf("%d", &position);

if (position >= n+1)

printf("Deletion not possible.\n");

else

{

for (c = position - 1; c < n - 1; c++)

array[c] = array[c+1];

printf("Resultant array:\n");

for (c = 0; c < n - 1; c++)

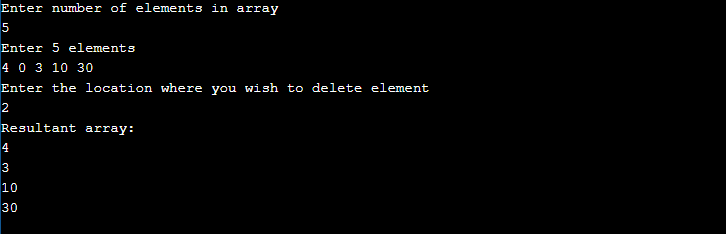
printf("%d\n", array[c]);

}

return 0;

}

**OUTPUT:**



**BAZGHA RAZI**

**Date Of Program**:02/09/2020 **Program Number**:03

**Roll Number**:

**AIM:** WAP to find the location of a given element in the array.

**PROGRAM:**

#include <stdio.h>

#define MAX\_SIZE 100 // Maximum array size

int main()

{ int arr[MAX\_SIZE];

int size, i, toSearch, found;

printf("Enter size of array: "); // Input size of array

scanf("%d", &size);

printf("Enter elements in array: "); //Input elements of array

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

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

}

printf("\nEnter element to search: ");

scanf("%d", &toSearch);

found = 0; //Assume that element does not exists in array

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

{ //If element is found in array then raise found flag and terminate from loop.

if(arr[i] == toSearch)

{

found = 1;

break;

}

} // If element is not found in array

if(found == 1)

{

printf("\n%d is found at position %d", toSearch, i + 1);

}

else

{

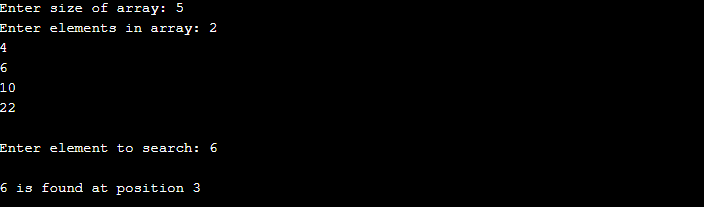
printf("\n%d is not found in the array", toSearch);

}

return 0;

}

**OUTPUT:**



**BAZGHA RAZI**

**Date Of Program**:07/09/2020 **Program Number**:04

**Roll Number**:

**AIM:** WAP to insert an element into a linked list.

**PROGRAM:**

#include <stdio.h>

#include <stdlib.h>

//Structure of a node

struct node {

int data; // Data

struct node \*next; // Address

}\*head;

void createList(int n);

void insertNodeAtBeginning(int data);

void displayList();

int main()

{

int n, data;

printf("Enter the total number of nodes: "); //Create a singly linked list of n nodes

scanf("%d", &n);

createList(n);

printf("\nData in the list \n");

displayList();

printf("\nEnter data to insert at beginning of the list: "); //Insert data at the beginning of the singly linked list

scanf("%d", &data);

insertNodeAtBeginning(data);

printf("\nData in the list \n");

displayList();

return 0;

}

void createList(int n) //Create a list of n nodes

{ struct node \*newNode, \*temp;

int data, i;

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

if(head == NULL) // If unable to allocate memory for head node

{ printf("Unable to allocate memory.");

}

else

{ printf("Enter the data of node 1: "); // Input data of node from the user

scanf("%d", &data);

head->data = data; // Link data field with data

head->next = NULL; // Link address field to NULL

temp = head;

for(i=2; i<=n; i++) // Create n nodes and adds to linked list

{

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

if(newNode == NULL) // If memory is not allocated for newNode

{ printf("Unable to allocate memory.");

break;

}

else

{ printf("Enter the data of node %d: ", i);

scanf("%d", &data);

newNode->data = data; // Link data field of newNode with data

newNode->next = NULL; // Link address field of newNode with NULL

temp->next = newNode; // Link previous node i.e. temp to the newNode

temp = temp->next;

}

}

printf("LINKED LIST CREATED SUCCESSFULLY\n");

}

}

void insertNodeAtBeginning(int data) // Create a new node and inserts at the beginning of the linked list.

{ struct node \*newNode;

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

if(newNode == NULL)

{ printf("Unable to allocate memory.");

}

else

{ newNode->data = data; // Link data part

newNode->next = head; // Link address part

head = newNode; // Make newNode as first node

printf("DATA INSERTED SUCCESSFULLY\n");

}

}

void displayList() // Display entire list

{ struct node \*temp;

if(head == NULL) // If the list is empty i.e. head = NULL

{

printf("List is empty.");

}

else

{ temp = head;

while(temp != NULL)

{ printf("Data = %d\n", temp->data); // Print data of current node

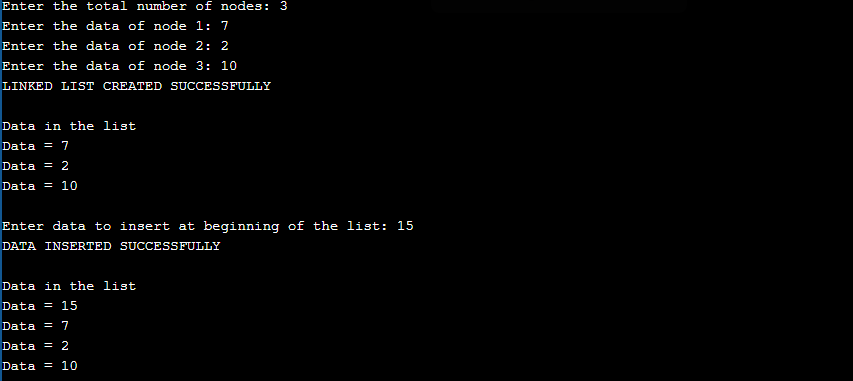
temp = temp->next; // Move to next node

}

}

}

**OUTPUT:**



**BAZGHA RAZI**

**Date Of Program**:09/09/2020 **Program Number**:05

**Roll Number**:

**AIM:** WAP to delete an element from the linked list.

**PROGRAM:**

#include <stdio.h>

#include <stdlib.h>

//Structure of a node

struct node {

int data; // Data

struct node \*next; // Address

}\*head;

void createList(int n);

void deleteFirstNode();

void displayList();

int main()

{ int n, choice;

printf("Enter the total number of nodes: "); //Create a singly linked list of n nodes

scanf("%d", &n);

createList(n);

printf("\nData in the list \n");

displayList();

printf("\nPress 1 to delete first node: ");

scanf("%d", &choice);

if(choice == 1) //Delete first node from list

deleteFirstNode();

printf("\nData in the list \n");

displayList();

return 0;

}

void createList(int n) //Create a list of n nodes

{ struct node \*newNode, \*temp;

int data, i;

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

if(head == NULL) //If unable to allocate memory for head node

{ printf("Unable to allocate memory.");

}

else

{ printf("Enter the data of node 1: "); //In data of node from the user

scanf("%d", &data);

head->data = data; // Link the data field with data

head->next = NULL; // Link the address field to NULL

temp = head;

for(i=2; i<=n; i++) //Create n nodes and adds to linked list

{ newNode = (struct node \*)malloc(sizeof(struct node));

if(newNode == NULL) //If memory is not allocated for newNode

{ printf("Unable to allocate memory.");

break;

}

else

{ printf("Enter the data of node %d: ", i);

scanf("%d", &data);

newNode->data = data; // Link the data field of newNode with data

newNode->next = NULL; // Link the address field of newNode with NULL

temp->next = newNode; // Link previous node i.e. temp to the newNode

temp = temp->next;

}

}

printf("SINGLY LINKED LIST CREATED SUCCESSFULLY\n");

}

}

void deleteFirstNode() //Deletes the first node of the linked list

{ struct node \*toDelete;

if(head == NULL)

{ printf("List is already empty.");

}

else

{ toDelete = head;

head = head->next;

printf("\nData deleted = %d\n", toDelete->data);

free(toDelete); //Clears the memory occupied by first node

printf("SUCCESSFULLY DELETED FIRST NODE FROM LIST\n");

}

}

void displayList() //Displays the entire list

{ struct node \*temp;

if(head == NULL) //If the list is empty i.e. head = NULL

{ printf("List is empty.");

}

else

{ temp = head;

while(temp != NULL)

{

printf("Data = %d\n", temp->data); // Print data of current node

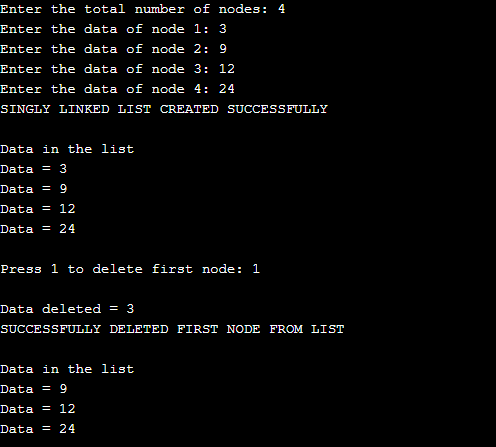
temp = temp->next; // Move to next node

}

}

}

**OUTPUT:**



**BAZGHA RAZI**

**Date Of Program**:10/09/2020 **Program Number**:06

**Roll Number**:

**AIM:** WAP to search and display element of a linked list.

**PROGRAM:**

#include <stdio.h>

#include <stdlib.h>

struct node

{ int num;

struct node \*nextptr;

}

stnode, \*ennode;

int FindElement(int);

void main()

{ int n,i,FindElem,FindPlc;

stnode.nextptr=NULL;

ennode=&stnode;

printf("\n\n Linked List : Search an element in a Linked List :\n");

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

printf(" Input the number of nodes : ");

scanf("%d", &n);

printf("\n");

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

{ ennode->nextptr=(struct node \*)malloc(sizeof(struct node));

printf(" Input data for node %d : ",i+1);

scanf("%d",&ennode->num);

ennode=ennode->nextptr;

}

ennode->nextptr=NULL;

printf("\n Data entered in the list are :\n");

ennode=&stnode;

while(ennode->nextptr!=NULL)

{ printf(" Data = %d\n",ennode->num);

ennode=ennode->nextptr;

}

printf("\n");

printf(" Input the element to be searched : ");

scanf("%d",&FindElem);

FindPlc=FindElement(FindElem);

if(FindPlc<=n)

printf(" Element found at node %d \n\n",FindPlc);

else

printf(" This element does not exists in linked list.\n\n");

}

int FindElement(int FindElem)

{ int ctr=1;

ennode=&stnode;

while(ennode->nextptr!=NULL)

{ if(ennode->num==FindElem)

break;

else

ctr++;

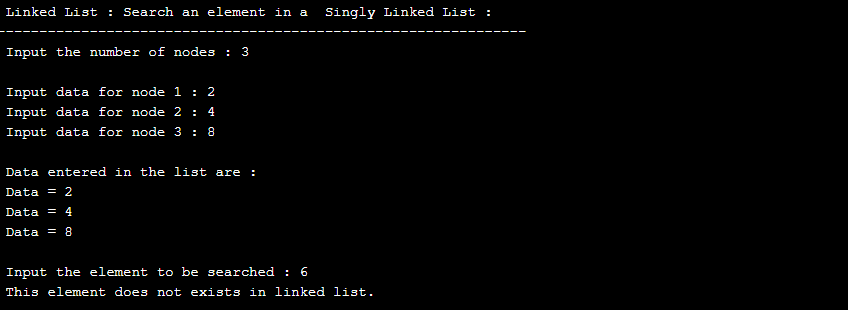
ennode=ennode->nextptr;

}

return ctr;

}

**OUTPUT:**



**BAZGHA RAZI**

**Date Of Program**:14/09/2020 **Program Number**:07

**Roll Number**:

**AIM:** WAP to implement all the stack operations:

-push

-pop

-display

**PROGRAM:**

#include<stdio.h>

#include<process.h>

#include<stdlib.h>

#define MAX 5 //Maximum number of elements that can be stored

int top=-1,stack[MAX];

void push();

void pop();

void display();

void main()

{ int ch;

while(1) //infinite loop, will end when choice will be 4

{ printf("\n\*\*\* Stack Menu \*\*\*");

printf("\n\n1.Push\n2.Pop\n3.Display\n4.Exit");

printf("\n\nEnter your choice(1-4):");

scanf("%d",&ch);

switch(ch)

{ case 1: push();

break;

case 2: pop();

break;

case 3: display();

break;

case 4: exit(0);

default: printf("\nWrong Choice!!");

}

}

}

void push()

{ int val;

if(top==MAX-1)

{ printf("\nStack is full!!");

}

else

{ printf("\nEnter element to push:");

scanf("%d",&val);

top=top+1;

stack[top]=val;

}

}

void pop()

{ if(top==-1)

{ printf("\nStack is empty!!");

}

else

{ printf("\nDeleted element is %d",stack[top]);

top=top-1;

}

}

void display()

{ int i;

if(top==-1)

{

printf("\nStack is empty!!");

}

else

{ printf("\nStack is...\n");

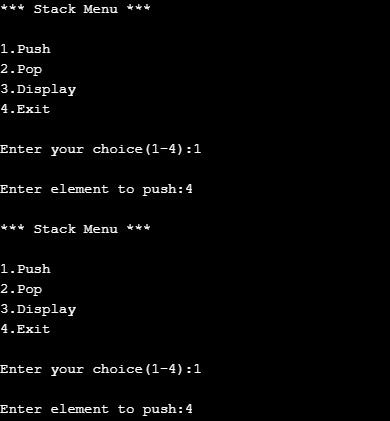
for(i=top;i>=0;--i)

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

}

}

**OUTPUT:**



**BAZGHA RAZI**

**Date Of Program**:16/09/2020 **Program Number**:08

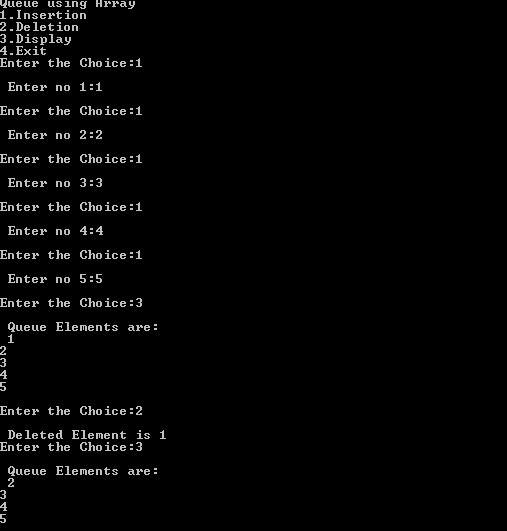
**Roll Number**:

**AIM:** WAP to implement various operations on a linear queue using linear array.

**PROGRAM:**

#include<stdio.h>  
#define n 5  
int main()  
{ int queue[n],ch=1,front=0,rear=0,i,j=1,x=n;  
    printf("Queue using Array");  
    printf("\n1.Insertion \n2.Deletion \n3.Display \n4.Exit");  
    while(ch)  
    { printf("\nEnter the Choice:");  
        scanf("%d",&ch);  
        switch(ch)  
        { case 1:  
            if(rear==x)  
                printf("\n Queue is Full");  
            else  
            { printf("\n Enter no %d:",j++);  
                scanf("%d",&queue[rear++]); }  
            break;  
        case 2:  
            if(front==rear)  
            { printf("\n Queue is empty"); }  
            else  
            {   printf("\n Deleted Element is %d",queue[front++]);  
                x++; }  
            break;  
        case 3:  
            printf("\n Queue Elements are:\n ");  
            if(front==rear)  
                printf("\n Queue is Empty");  
            else  
            {  
                for(i=front; i<rear; i++)  
                {  
                    printf("%d",queue[i]);  
                    printf("\n");  
                }  
                break;  
            case 4:  
                exit(0);  
            default:  
                printf("Wrong Choice: please see the options");  
            }  
        }  
    }  
    return 0;  
}

**OUTPUT:**



**BAZGHA RAZI**

**Date Of Program**:22/09/2020 **Program Number**:09

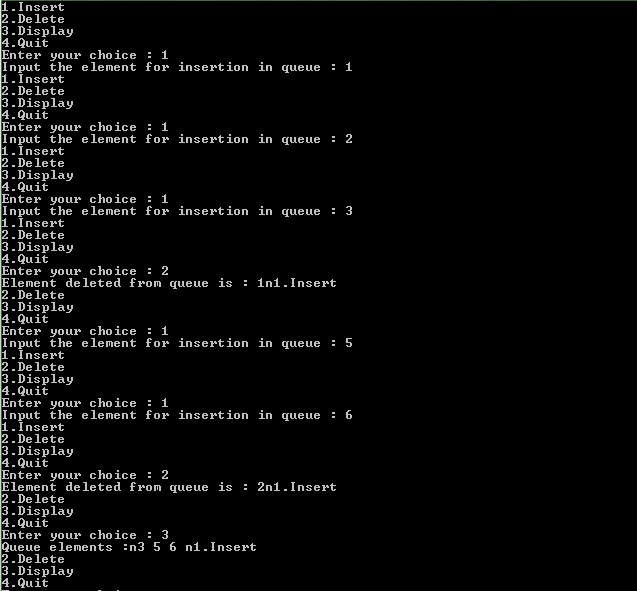
**Roll Number**:

**AIM:** WAP to implement various operations on a circular queue using linear array.

**PROGRAM:**

#include<stdio.h>  
# define MAX 5  
int cqueue\_arr[MAX];  
int front = -1;  
int rear = -1;  
void insert(int item)  
{ if((front == 0 && rear == MAX-1) || (front == rear+1))  
{ printf("Queue Overflow n");  
return;  
}  
if(front == -1)  
{ front = 0;  
rear = 0; }  
else  
{ if(rear == MAX-1)  
rear = 0;  
else  
rear = rear+1; }  
cqueue\_arr[rear] = item ;  
}  
void deletion()  
{ if(front == -1)  
{ printf("Queue Underflown");  
return ;  
}  
printf("Element deleted from queue is : %dn",cqueue\_arr[front]);  
if(front == rear)  
{ front = -1;  
rear=-1;  
}  
else  
{ if(front == MAX-1)  
front = 0;  
else  
front = front+1; }  
}  
void display()  
{ int front\_pos = front,rear\_pos = rear;  
if(front == -1)  
{ printf("Queue is emptyn");  
return; }  
printf("Queue elements :n");  
if( front\_pos <= rear\_pos )  
while(front\_pos <= rear\_pos)  
{ printf("%d ",cqueue\_arr[front\_pos]);  
front\_pos++; }  
else  
{ while(front\_pos <= MAX-1)  
{ printf("%d ",cqueue\_arr[front\_pos]);  
front\_pos++; }  
front\_pos = 0;  
while(front\_pos <= rear\_pos)  
{ printf("%d ",cqueue\_arr[front\_pos]);  
front\_pos++; }  
}  
printf("n");  
}  
int main()  
{ int choice,item;  
do  
{ printf("1.Insert\n");  
printf("2.Delete\n");  
printf("3.Display\n");  
printf("4.Quit\n");  
printf("Enter your choice : ");  
scanf("%d",&choice);  
switch(choice)  
{  
case 1 :  
printf("Input the element for insertion in queue : ");  
scanf("%d", &item);  
insert(item);  
break;  
case 2 :  
deletion();  
break;  
case 3:  
display();  
break;  
case 4:  
break;  
default:  
printf("Wrong choicen");  
}  
}while(choice!=4);  
return 0;  
}

**OUTPUT:**



**BAZGHA RAZI**

**Date Of Program**:23/09/2020 **Program Number**:10

**Roll Number**:

**AIM:** WAP to implement various operations on a queue using linear linked list.

**PROGRAM:**

#include<stdio.h>

#include<stdlib.h>

struct node

{

int data;

struct node \*next;

};

struct node \*front;

struct node \*rear;

void insert();

void delete();

void display();

void main ()

{

int choice;

while(choice != 4)

{ printf("\n1.insert an element\n2.Delete an element\n3.Display the queue\n4.Exit\n");

printf("\nEnter your choice ?");

scanf("%d",& choice);

switch(choice)

{

case 1:

insert();

break;

case 2:

delete();

break;

case 3:

display();

break;

case 4:

exit(0);

break;

default:

printf("\nEnter valid choice??\n");

}

}

}

void insert()

{

struct node \*ptr;

int item;

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

if(ptr == NULL)

{

printf("\nOVERFLOW\n");

return;

}

else

{

printf("\nEnter value?\n");

scanf("%d",&item);

ptr -> data = item;

if(front == NULL)

{

front = ptr;

rear = ptr;

front -> next = NULL;

rear -> next = NULL;

}

else

{

rear -> next = ptr;

rear = ptr;

rear->next = NULL;

}

}

}

void delete ()

{

struct node \*ptr;

if(front == NULL)

{

printf("\nUNDERFLOW\n");

return;

}

else

{

ptr = front;

front = front -> next;

free(ptr);

}

}

void display()

{

struct node \*ptr;

ptr = front;

if(front == NULL)

{

printf("\nEmpty queue\n");

}

else

{ printf("\nprinting values .....\n");

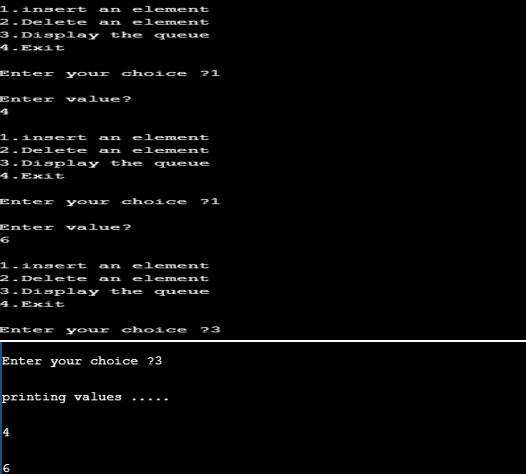
while(ptr != NULL)

{

printf("\n%d\n",ptr -> data);

ptr = ptr -> next; } } }

**OUTPUT:**



**BAZGHA RAZI**

**Date Of Program**:28/09/2020 **Program Number**:11

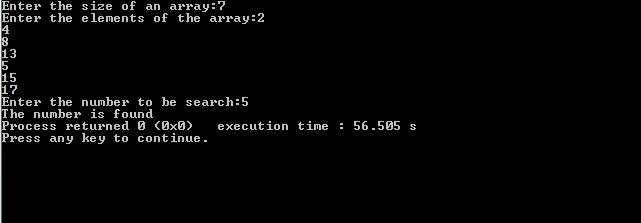
**Roll Number**:

**AIM:** WAP to demonstrate the use of linear search of a given element in an array.

**PROGRAM:**

#include<stdio.h>  
int main()  
{  
int a[10],i,n,m,c=0;  
printf("Enter the size of an array:");  
scanf("%d",&n);  
printf("Enter the elements of the array:");  
for(i=0;i<=n-1;i++)  
{  
scanf("%d",&a[i]);  
}  
printf("Enter the number to be search:");  
scanf("%d",&m);  
for(i=0;i<n;i++)  
{  
if (a[i]==m)  
{  
c=1;  
break;  
}  
}  
if(c==0)  
printf("The number is not in the list");  
else  
printf("The number is found");  
return 0;  
}

**OUTPUT:**

****

**BAZGHA RAZI**

**Date Of Program**:05/10/2020 **Program Number**:12

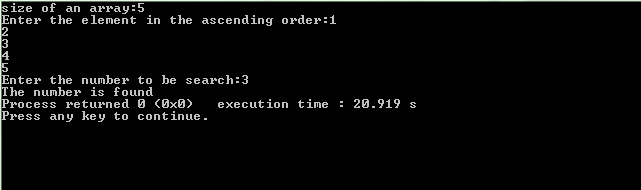
**Roll Number**:

**AIM:** WAP to demonstrate the use of binary search of a given element in an array.

**PROGRAM:**

#include<stdio.h>  
int main()  
{  
int a[10],i,n,m,c=0,l,u,mid;  
printf("size of an array:");  
scanf("%d",&n);  
printf("Enter the element in the ascending order:");  
for(i=0;i<n;i++)  
{  
scanf("%d",&a[i]);  
}  
printf("Enter the number to be search:");  
scanf("%d",&m);  
l=0,u=n-1;  
while(l<=u)  
{  
mid=(l+u)/2;  
if(m==a[mid])  
{  
c=1;  
break;  
}  
 else if(m<a[mid])  
{  
u=mid-l;  
}  
else  
l=mid+1;  
}  
if(c==0)  
printf("The number is not found");  
else  
printf("The number is found");  
return 0;  
}

**OUTPUT:**



**BAZGHA RAZI**

**Date Of Program**:13/10/2020 **Program Number**:13

**Roll Number**:

**AIM:** WAP to illustrate the implementation of different operations on a BST.

**PROGRAM:**

#include <stdio.h>

#include <stdlib.h>

struct node {​​​​​​

  int item;

  struct node\* left;

  struct node\* right;

}​​​​​​;

// Inorder traversal

void inorderTraversal(struct node\* root) {​​​​​​

  if (root == NULL) return;

  inorderTraversal(root->left);

  printf("%d ->", root->item);

  inorderTraversal(root->right);

}​​​​​​

// preorderTraversal traversal

void preorderTraversal(struct node\* root) {​​​​​​

  if (root == NULL) return;

  printf("%d ->", root->item);

  preorderTraversal(root->left);

  preorderTraversal(root->right);

}​​​​​​

// postorderTraversal traversal

void postorderTraversal(struct node\* root) {​​​​​​

  if (root == NULL) return;

  postorderTraversal(root->left);

  postorderTraversal(root->right);

  printf("%d ->", root->item);

}​​​​​​

// Create a new Node

struct node\* createNode(value) {​​​​​​

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

  newNode->item = value;

  newNode->left = NULL;

  newNode->right = NULL;

  return newNode;

}​​​​​​

// Insert on the left of the node

struct node\* insertLeft(struct node\* root, int value) {​​​​​​

  root->left = createNode(value);

  return root->left;

}​​​​​​

 // Insert on the right of the node

struct node\* insertRight(struct node\* root, int value) {​​​​​​

  root->right = createNode(value);

  return root->right;

}​​​​​​

int main() {​​​​​​

  struct node\* root = createNode(1);

  insertLeft(root, 12);

  insertRight(root, 9);

  insertLeft(root->left, 5);

  insertRight(root->left, 6);

  printf("Inorder traversal \n");

  inorderTraversal(root);

  printf("\nPreorder traversal \n");

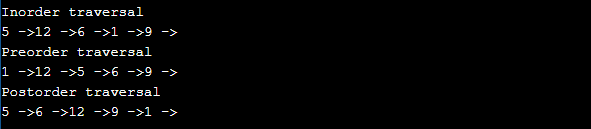
  preorderTraversal(root);

  printf("\nPostorder traversal \n");

  postorderTraversal(root);

}​​​​​​

**OUTPUT:**



**BAZGHA RAZI**

**Date Of Program**:23/10/2020 **Program Number**:14

**Roll Number**:

**AIM:** WAP to sort an array of integers in ascending order using bubble sort.

**PROGRAM:**

#include <stdio.h>

int main()

{

  int array[100], n, c, d, swap;

  printf("Enter number of elements**\n**");

  scanf("%d", &n);

  printf("Enter %d integers**\n**", n);

  for (c = 0; c < n; c++)

    scanf("%d", &array[c]);

  for (c = 0 ; c < n - 1; c++)

  {

    for (d = 0 ; d < n - c - 1; d++)

    {

      if (array[d] > array[d+1]) */\* For decreasing order use '<' instead of '>' \*/*

      {

        swap       = array[d];

        array[d]   = array[d+1];

        array[d+1] = swap;

      }

    }

  }

  printf("Sorted list in ascending order:**\n**");

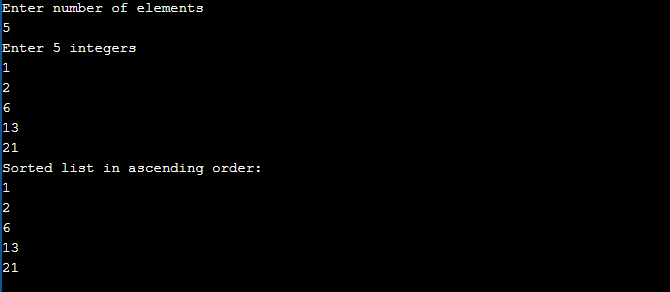
  for (c = 0; c < n; c++)

     printf("%d**\n**", array[c]);

  return 0;

}

**OUTPUT:**



**BAZGHA RAZI**

**Date Of Program**:02/11/2020 **Program Number**:15

**Roll Number**:

**AIM:** WAP to sort an array of integers in ascending order using selection sort.

**PROGRAM:**

#include<stdio.h>

int main(){

  /\* Here i & j for loop counters, temp for swapping,

   \* count for total number of elements, number[] to

   \* store the input numbers in array. You can increase

   \* or decrease the size of number array as per requirement

   \*/

  int i, j, count, temp, number[25];

  printf("How many numbers u are going to enter?: ");

  scanf("%d",&count);

  printf("Enter %d elements: ", count);

  // Loop to get the elements stored in array

  for(i=0;i<count;i++)

     scanf("%d",&number[i]);

  // Logic of selection sort algorithm

  for(i=0;i<count;i++){

     for(j=i+1;j<count;j++){

        if(number[i]>number[j]){

           temp=number[i];

           number[i]=number[j];

           number[j]=temp;

        }

     }

  }

  printf("Sorted elements: ");

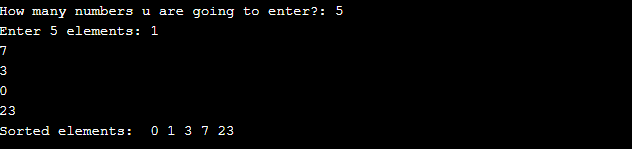
  for(i=0;i<count;i++)

     printf(" %d",number[i]);

  return 0;

}

**OUTPUT:**



**BAZGHA RAZI**

**Date Of Program**:09/11/2020 **Program Number**:16

**Roll Number**:

**AIM:** WAP to sort an array of integers in ascending order using insertion sort.

**PROGRAM:**

#include<stdio.h>

int main(){

/\* Here i & j for loop counters, temp for swapping,

\* count for total number of elements, number[] to

\* store the input numbers in array. You can increase

\* or decrease the size of number array as per requirement

\*/

int i, j, count, temp, number[25];

printf("How many numbers u are going to enter?: ");

scanf("%d",&count);

printf("Enter %d elements: ", count);

// This loop would store the input numbers in array

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

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

// Implementation of insertion sort algorithm

for(i=1;i<count;i++){

temp=number[i];

j=i-1;

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

number[j+1]=number[j];

j=j-1;

}

number[j+1]=temp;

}

printf("Order of Sorted elements: ");

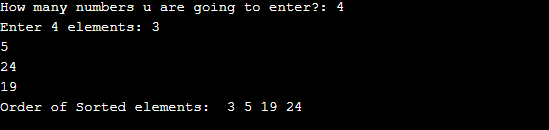
for(i=0;i<count;i++)

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

return 0;

}

**OUTPUT:**



**BAZGHA RAZI**

**Date Of Program**:01/12/2020 **Program Number**:17

**Roll Number**:

**AIM:** WAP to sort an array of integers in ascending order using merge sort.

**PROGRAM:**

#include<stdio.h>

void mergesort(int a[],int i,int j);

void merge(int a[],int i1,int j1,int i2,int j2);

int main()

{

int a[30],n,i;

printf("Enter no of elements:");

scanf("%d",&n);

printf("Enter array elements:");

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

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

mergesort(a,0,n-1);

printf("\nSorted array is :");

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

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

return 0;

}

void mergesort(int a[],int i,int j)

{

int mid;

if(i<j)

{

mid=(i+j)/2;

mergesort(a,i,mid); //left recursion

mergesort(a,mid+1,j); //right recursion

merge(a,i,mid,mid+1,j); //merging of two sorted sub-arrays

}

}

void merge(int a[],int i1,int j1,int i2,int j2)

{

int temp[50]; //array used for merging

int i,j,k;

i=i1; //beginning of the first list

j=i2; //beginning of the second list

k=0;

while(i<=j1 && j<=j2) //while elements in both lists

{

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

temp[k++]=a[i++];

else

temp[k++]=a[j++];

}

while(i<=j1) //copy remaining elements of the first list

temp[k++]=a[i++];

while(j<=j2) //copy remaining elements of the second list

temp[k++]=a[j++];

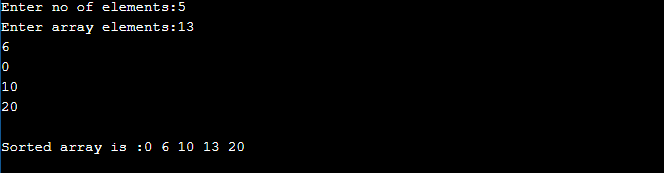
//Transfer elements from temp[] back to a[]

for(i=i1,j=0;i<=j2;i++,j++)

a[i]=temp[j];

}

**OUTPUT:**



**BAZGHA RAZI**

**Date Of Program**:14/12/2020 **Program Number**:18

**Roll Number**:

**AIM:** WAP to sort an array of integers in ascending order using quick sort.

**PROGRAM:**

#include<stdio.h>

void quicksort(int number[25],int first,int last){​​

int i, j, pivot, temp;

if(first<last){​​

pivot=first;

i=first;

j=last;

while(i<j){​​

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

i++;

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

j--;

if(i<j){​​

temp=number[i];

number[i]=number[j];

number[j]=temp;

}​​

}​​

temp=number[pivot];

number[pivot]=number[j];

number[j]=temp;

quicksort(number,first,j-1);

quicksort(number,j+1,last);

}​​

}​​

int main(){​​

int i, count, number[25];

printf("How many elements are u going to enter?: ");

scanf("%d",&count);

printf("Enter %d elements: ", count);

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

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

quicksort(number,0,count-1);

printf("Order of Sorted elements: ");

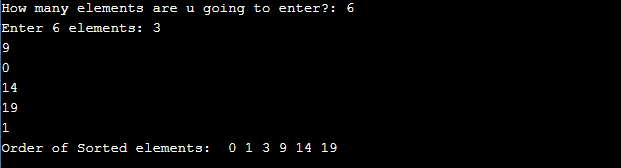
for(i=0;i<count;i++)

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

return 0;

}​​

**OUTPUT:**



**BAZGHA RAZI**

**Date Of Program**:22/12/2020 **Program Number**:19

**Roll Number**:

**AIM:** WAP to sort an array of integers in ascending order using heap sort.

**PROGRAM:**

#include<stdio.h>

void create(int []);

void down\_adjust(int [],int);

void main()

{ int heap[30],n,i,last,temp;

printf("Enter no. of elements:");

scanf("%d",&n);

printf("\nEnter elements:");

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

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

//create a heap

heap[0]=n;

create(heap);

//sorting

while(heap[0] > 1)

{

//swap heap[1] and heap[last]

last=heap[0];

temp=heap[1];

heap[1]=heap[last];

heap[last]=temp;

heap[0]--;

down\_adjust(heap,1);

}

//print sorted data

printf("\nArray after sorting:\n");

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

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

}

void create(int heap[])

{

int i,n;

n=heap[0]; //no. of elements

for(i=n/2;i>=1;i--)

down\_adjust(heap,i);

}

void down\_adjust(int heap[],int i)

{

int j,temp,n,flag=1;

n=heap[0];

while(2\*i<=n && flag==1)

{ j=2\*i; //j points to left child

if(j+1<=n && heap[j+1] > heap[j])

j=j+1;

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

flag=0;

else

{ temp=heap[i];

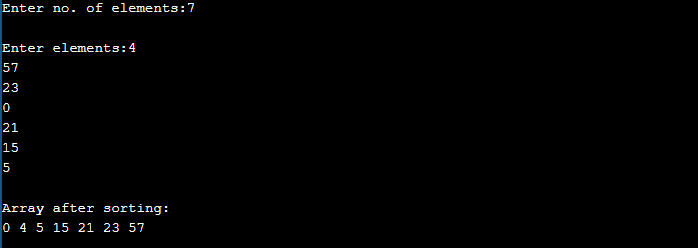
heap[i]=heap[j];

heap[j]=temp;

i=j;

} } }

**OUTPUT:**



**BAZGHA RAZI**

**Date Of Program**:29/12/2020 **Program Number**:20

**Roll Number**:

**AIM:** WAP to sort an array of integers in ascending order using shell sort.

**PROGRAM:**

#include<stdio.h>

void sort(int a[],int n)

{

int gap,i,j,temp;

for(gap=n/2;gap>0;gap/=2)

{

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

{

temp=a[i];

for(j=i;j>=gap&&a[j-gap]>temp;j-=gap)

a[j]=a[j-gap];

a[j]=temp;

}

}

}

int main()

{

int a[20],i,n;

printf("Enter number of elements:");

scanf("%d",&n);

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

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

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

sort(a,n);

printf("\nArray after shell sort:\n");

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

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

    return 0;

}

**OUTPUT:**

