AIM: Write a C-program to search whether an item K present in an array of N elements (Using linear and binary search algorithms)

Constraints: $1 < K < 1000$ $1 < N < 1000$

ALGORITHM:

1. Start
2. Declare array of size N and items.
3. Read data (list of elements into array) and read the items.
4. Compare the key element with the items present in the array.
5. If the key element is i.e item then print Item is found and exit (or) else print item is not found.
6. Stop

DESCRIPTION:

Eg.: i) 4 32 73 12 14

item = 32

item is found

ii) 4 32 73 12 14

item = 90

item is not found

PROGRAM:

```
#include <stdio.h>
#define N 10
int main()
{
  int arr[N];
  int item, i;
  for(i=0; i<N; i++)
  {
    scanf("%d", &arr[i]);
```

```
}
scanf("%d", &item);
if ((N>1 && N<1000) && (item>1 && item<1000))
{
  for(i=0; i<N; i++)
  {
    if (item == arr[i])
    {
      printf("item found at %d", i);
      break;
    }
  }
  if (i==N)
  {
    printf("item is not found");
  }
}
else
{
  printf("item should be <1000");
}
}
```

OUTPUT:

Test Case 1: 45 78 123 48 34 89 67 54 74 543

Key: 34 enter item to find: 34

item found at 4

Test Case 2: 45 78 123 48 34 89 67 54 74 543 Key: 343

enter item to find: 343 item not found at

AIM: To search an element in an array using Binary search.

ALGORITHM:

1. Start
2. Declare i, j, a[N], temp, sorted[10], l, r, temp, mid, key.
3. Read size[N] and elements in the array.
4. Sort the elements into the array a[N].
5. Read the sorted elements into another array sorted[10].
6. Print the sorted array sorted[10]
7. Initialise $l=0$, $r=N-1$.
8. Give the condition for constraints.

   $1<key<1000$ $\quad$ $1<N<1000$.
9. If key == sorted[mid], print key is found
   else if key < sorted[mid] $\quad$ $r=mid-1$
   else $\quad$ $l=mid+1$
10. If ($l>r$) print key is not found.
11. If key is above 1000, print entered key is more than 1000.
12. Stop

DESCRIPTION:

Eg.: 5 10 15 20 25

element to find: 20

	5	10	15	20	25
i=	0	1	2	3	4

$$mid = \frac{0+4}{2} = 2$$

the element at mid is 15 $\quad$ $15<20$

if (arr[mid] < key) $\quad$ $l=mid+1$

$l=3$

the element is found at 3.

PROGRAM:

```
#include<stdio.h>
#define N 10
int main()
{
  int i, j, a[N], temp, sorted[10];
  for(i=0; i<N, i++)
  {
    scanf("%d", &a[i]);
  }
  for(i=0; i<N-1; i++)
  {
    for(j=0; j<N-1-i; j++)
    {
      if(a[j] > a[j+1])
      {
        temp = a[j+1];
        a[j+1] = a[j];
        a[j] = temp;
      }
    }
  }
  for(i=0; i<N; i++)
  {
    sorted[i] = a[i];
  }
  for(j=0; j<N; j++)
  {
```

```
    printf("%d", a[j]);
}
printf("\n");
for(i=0; i<N; i++)
{
  printf("%d", sorted[i]);
}
int l=0, r=N-1, key, mid;
printf("enter the key elemint:");
scanf("%d", &key);
if((key>1)&&(key<1000)&&(N>1)&&(N<1000))
{
  while(l<=r)
  {
    mid = (l+r)/2;
    if(key == sorted[mid])
    {
      printf("key is found");
      break;
    }
  else if(key < sorted[mid])
    {
      r = mid-1;
    }
  else if(key > sorted[mid])
    {
      l = mid+1;
    }
```

```
if (l>r)
  printf(" key not found");
}
else
  printf (" entered key is more than 1000");
}
```

OUTPUT:

Test Case 1:

input array: 45 78 123 48 34 89 67 54 74 543.

34 45 48 58 67 74 78 89 123 543

enter the key element : 34

key is found.

Test Case 2:

input array: 45 78 123 48 34 89 67 54 74 543

34 45 48 58 67 74 78 89 123 543

enter the key element : 542

key not found.

AIM: Write a program to sort the given array of N elements using divide and conquer method (merge sort)

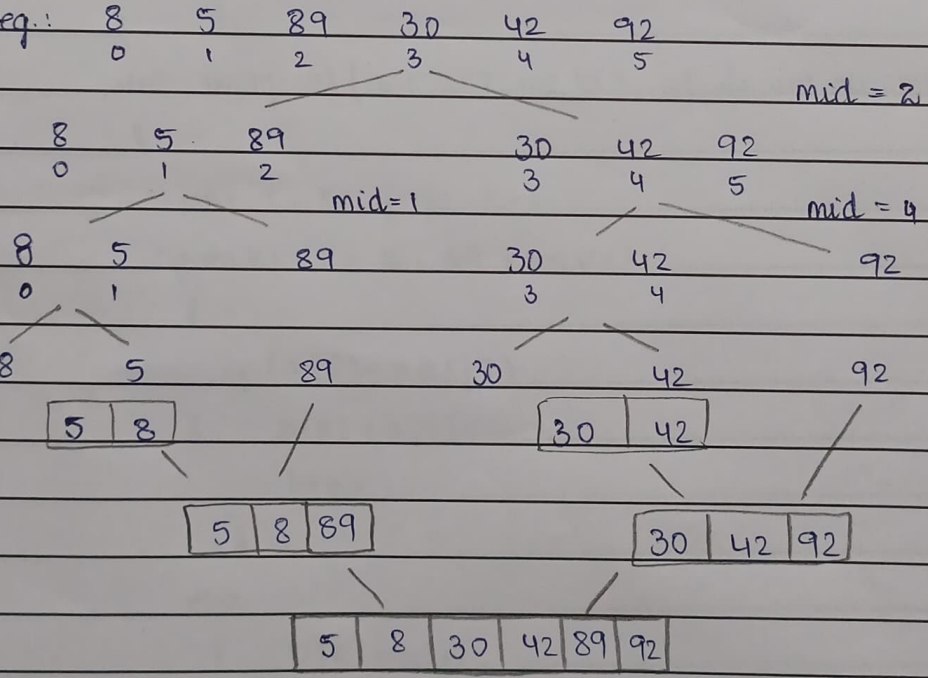
Constraints: $1 < N < 1000$

ALGORITHM:

1. Start
2. declare array size, array and required variables
3. read the value of array size and elements into array.
4. divide the array into 2 halves until the element becomes single.
5. From the single element sort and merge the element
6. copy the sorted elements into the original array
7. print the final array
8. Stop

DESCRIPTION:

eg.:



The sorted array is 5 8 30 42 89 92

PROGRAM:

```
#include<stdio.h>
int mergesort(int a[], int l, int u)
{
  int b[100], mid;
  if (l<u)
  {
    mid = (l+u)/2;
    mergesort(a,l,mid);
    mergesort(a,mid+1,u);
    mergesort(a,b,l,mid,mid+1,u);
    copy(a,b,l,u);
  }
}
int mergesort(int a[],int b[], int l1, int u1, int l2, int u2)
{
  int i=l1, j=l2, k=l1;
  while(i<=u1 && j<=u2)
  {
    if(a[i]<=a[j])
    {   b[k]=a[i];
        i++;
    }
  else
    {   b[k]=a[j];
        j++;
    }
    k++;   }
```

```
while(i<=u1)
{   temp[k++]=a[i++];
}
while(j<=u2)
{  temp[k++]=a[j++];
}
int copy(int a[], int b[], int l, int u)
{   int i;
    for(i=l; i<=u; i++)
       a[i]=b[i];
 }
void main()
{     int a[100],n,i;
      printf("enter array size:");
      scanf("%d",&n);
      printf("enter the elements into the array");
      for(i=0; i<n; i++)
       scanf("%d",&a[i]);
      printf("before sorting");
      for(i=0; i<n; i++)
      printf("%d", a[i]);
      printf("after sorting");
      for(i=0; i<n; i++)
      printf("%d", a[i]);
    }
```

Output:

Test case 1: enter array size: 15

enter the elements into array

87 36 9 12 24 5 78 567 456 34 96 45 39 89 123

Before sorting

87 36 9 12 24 5 78 567 456 34 96 45 39 89 123

after sorting

5 9 12 24 34 36 39 45 78 87 89 96 123 456 567

Test case 2: enter array size: 8

enter the elements into array

4 8 5 89 30 42 92 70

before sorting

4 8 5 89 30 42 92 70

after sorting

4 5 8 30 42 70 89 92

PRAGNANAM BRAHMA
ANITS

SHEET No. _____

AIM: Design, Develop and Implement a menu driven program in C for the following.

a. Operations on STACK of integers

1. Push an element on to stack
2. Pop an element from stack
3. Demonstrate overflow and Underflow situations on stack
4. Display the status of stack
5. Exit

ALGORITHM:

1. Start
2. Mention header files, declare global variables MAX, top = -1, size of array.
3. Write push function. Read the item, print if it is overflow else increment top, and put it on top.
4. Write pop function. If there are no elements print it is underflow else decrement.
5. Write peek function. if there are no elements print it is underflow else print top element
6. Write display function. print if stack is underflow or else print the elements in array.
7. Declare main function. print choice and read it. perform the operations based on choice
8. Stop

DESCRIPTION:

top=-1 [| | |]

push(5) [5 | |] top = 0

push(10) [5 | 10 |] top = 1

push(15) [5 | 10 | 15] top = 2

push(20) overflow

pop(15) [5 | 10 |] top = 1

peek( ) [5 | 10 |] 10

display( ) 10 5

PROGRAM:

```
#include <stdio.h>
#include<stdlib.h>
#define MAX 3
int stack[MAX];
int top=-1;
int main()
{
  void push();
  void pop();
  void peek();
  void display();
  int choice, item;
  printf("stack operation\n");
  printf("1.push\n 2.pop\n 3.peek\n 4.display\n 5.exit\n");
  while(1)
  {
   printf("enter your choice");
   scanf("%d",&choice);
   switch(choice)
   {
    case 1: printf("enter item");
            scanf("%d",&item);
            push(item);
            break;
    case 2: pop();
            break;
    case 3: peek()
            break;
```

```
case 4: display();
        break;
case 5: exit(1);
        break;
default: printf("enter valid choice");
} }
return 0;
}
void push(int item)
{
  if (top == max-1)
  { printf("stack is overflow");
  }
  else
  { top++;
    stack[top]=item;
} }
void pop()
{ if (top==-1)
  { printf("stack is underflow");
  }
  else
  { printf("The element deleted is %d", stack[top]);
    top--;
} }
void peek()
{ if (top==-1)
  { printf("stack is underflow");
```

```
    }
    else
    { printf("top most element is %d", stack[top]);
    } }
    void display()
    {   if(top==-1)
        { printf("stack is underflow");
        }
        else
        { int i;
          printf("stack element are %d", stack[i]);
          for(i=top; i>=0; i--)
          { printf("%d ", stack[i]);
        } } }
```

OUTPUT:

Stack Operations:

1. push
2. pop
3. peek
4. display
5. exit

enter choice: 1
enter item: 5
enter choice: 1
enter item: 10
enter choice: 1
enter item: 15
enter choice: 2
The element deleted is 15
enter choice: 3
The top most element is 10
enter choice: 4
Stack elements are 10 5
enter choice: 5.