BINARY SEARCH

BINARY SEARCH ITERATIVE

```
/* Binary search using iterative */

#include <stdio.h>
#include <stdlib.h>
int binary_search(int A[],int l,int r,int x)    //function for binary search
 {
    int m;
    while(l<=r)
   {
      m=(l+r)/2;
      if(x==A[m])
         return m;
      if(x<A[m])
       {
         r=m-1;
       }
      else
       {
          l=m+1;
       }
   }
   return -1;
}

void sort(int A[],int n)     //function for sorting the random numbers using insertion sort
{
   int x,j;
   for(int i=2;i<=n;i++)
   {
      x=A[i];
      for( j=i-1;j>=1;j--)
      {
         if(x<A[j])
          {
```

```
            A[j+1]=A[j];
          }
          else
            break;
       }
       A[j+1]=x;
   }
}

int main(int args, char *argv[] )  //main function with command line argument
{
   int A[500],upper,lower,l,r,x,temp;
   lower=1,upper=1000;          // putting random value range
   l=1,r=500;         //putting values for left and right index

   for(int i = 1; i<=500; i++)   //taking random values
   {
     A[i]=(rand()%(upper - lower + 1)) + lower;
   }
   sort(A,500);    //function call for sorting
    x=atoi(argv[1]);   //taking element to be searched from command line argument

   temp=binary_search(A,l,r,x);   //function call for binary search
   if (temp!=-1)
   {
      printf("\nEntered value %d is present in %d",x,temp);
   }
   else
   {
      printf("\nValue is not present");
   }

   return 0;
}
```

BINARY SEARCH RECURSIVE

```
 /* Binary search using recursive */


#include <stdio.h>
#include <stdlib.h>
int binary_search_recursive(int A[],int l,int r,int x)     //function for binary search
{
   int m;
   if(l<=r)
   {
      m=(l+r)/2;
      if(A[m]==x)
         return m;
      if(x<A[m])
         return binary_search_recursive(A,l,m-1,x);
      return binary_search_recursive(A,m+1,r,x);
   }
   return -1;
}


void sort(int A[],int n)  //function for sorting the random numbers using insertion sort
{
   int x,j;
   for(int i=2;i<=n;i++)
   {
      x=A[i];
      for( j=i-1;j>=1;j--)
      {
         if(x<A[j])
         {
            A[j+1]=A[j];
         }
         else
            break;
      }
```

```
      A[j+1]=x;
   }
}

int main(int args, char *argv[])    //main function with command line argument
{
   int A[500],upper,lower,l,r,x,temp;
   lower=1,upper=1000;     // putting random value range
   l=1,r=500;       //puting values for left and right index

   for(int i = 1; i<=500; i++)  //taking random values
   {
     A[i]=(rand()%(upper - lower + 1)) + lower;
   }
   sort(A,500);  //function call for sorting
    x=atoi(argv[1]);   //taking element to be searched from command line argument

   temp=binary_search_recursive(A,l,r,x);  //function call for binary search
   if (temp!=-1)
   {
     printf("\nEntered value %d is present in %d",x,temp);
   }
   else
   {
     printf("\nValue is not present");
   }

   return 0;
}
```

COMPARISON

TYPE	BINARY SEARCH USING ITERATIVE	BINARY SEARCH USING RECURSIVE
Value Found	0.042 s	0.062 s
Value Not Found	0.051 s	0.046 s

There is not much time difference between the two .

FIBONACCI SERIES

FIBONACCI SERIES ITERATIVE

```
/* Fibonacci iterative */

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

void fibonacci_dp(int n)       //function Fibonacci  iterative
{
   int f[100],i;
   f[1]=0;
   f[2]=1;
   for(i=3;i<=n;i++)
   {
      f[i]=f[i-1]+f[i-2];
   }

   printf("%d\t",f[n]);     //printing nth fibonacci
}
int main(int args, char *argv[])     //main function with command line argument
{
   int n;
   n=atoi(argv[1]);    //taking nth value from command line argument
   printf("nth Fibonacci series\n");

   fibonacci_dp(n);    //function call for fibonacci iterative
}
```

FIBONACCI SERIES RECURSIVE

```
/* Fibonacci recursive */


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


int fibonacci_recursive(int n)    //function fibonacci recursive
{
   if(n==1)
      return 0;
   if(n==2)
      return 1;
   return fibonacci_recursive(n-1)+fibonacci_recursive(n-2);
}
int main(int args, char *argv[])  //main function with command line argument
{
   int n;
   n=atoi(argv[1]);  //taking nth value from command line argument
   printf("nth Fibonacci series\n");


   printf("%d\t",fibonacci_recursive(n));  //printing nth fibonacci and calling function
}
```

COMPARISON

VALUES	FIBONACCI ITERATIVE	FIBONACCI RECURSIVE
30	0.000 s	0.000 s
35	0.000 s	0.031 s
40	0.000 s	0.328s
45	0.0003 s	3.642 s
47	0.00031 s	9.534 s
49	0.00042 s	25.04 s
50	0.00047 s	40.34 s
51	0.0005 s	65.37 s

There is much difference between both the run times. Because of the recursion function Fibonacci recursive takes much time to execute than the Fibonacci iterative.