```
PRN: S17111009
Roll No.: 06
Class: BE Comp SS
Code (Binary Search):
#include<iostream>
#include<algorithm>
#include<stdlib.h>
#include<omp.h>
#include<ctime>
using namespace std;
int binary(int *, int, int, int);
int binary(int *a, int low, int high, int key)
{
     int mid;
     mid=(low+high)/2;
     int low1,low2,high1,high2,mid1,mid2,found=0,loc=1;
     #pragma omp parallel sections
     {
        #pragma omp section
           {
                 low∃ow;
                 high1=mid;
```

Name: Manasi. B. Kshirsagar

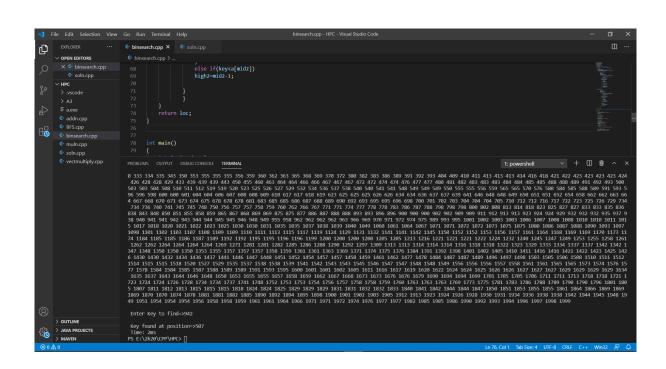
```
while(low1<=high1)
                  {
                        if(!(key>=a[low1] && key<=a[high1]))
                        {
                              low1=low1+high1;
                              continue;
                        }
                        mid1=(low1+high1)/2;
                        if(key=a[mid1])
                        {
                              found=1;
                              loc=mid1;
                              low1=high1+1;
                        }
                        else if(key>a[mid1])
                        {
                              low1=mid1+1;
                        else if(key<a[mid1])
                              high1=mid1-1;
                  }
            }
#pragma omp section
            {
```

```
low2=mid+1;
     high2=high;
     while(low2<=high2)
     {
           if(!(key>=a[low2] \&\& key<=a[high2]))
           {
                 low2=low2+high2;
                 continue;
           }
           mid2=(low2+high2)/2;
           if(key=a[mid2])
           {
                 found=1;
                 loc=mid2;
                 low2=high2+1;
           }
           else if(key>a[mid2])
           {
           low2=mid2+1;
           }
           else if(key<a[mid2])
           high2=mid2-1;
```

}

```
}
      }
      return loc;
}
int main()
{
      int *a,i,n,key,loc=1;
      cout<<"\n Enter Total no of Elements=>";
      cin>>n;
      a=new int[n];
      for(i=0;i<n;i++)
      {
       a[i]=rand()%(2*n);
  }
      sort(a,a+n);
      cout<<"Elements are: "<<endl;
      for(i=0;i<n;i++)
      {
       cout<<a[i]<<" ";
  }
      cout<<endl;
      clock_t start=clock();
      cout<<"\n Enter Key to find⇒";
```

## Output:



```
Code (DFS):
#include<iostream>
#include<omp.h>
#include<chrono>
#include<time.h>
using namespace std;
using namespace std::chrono;
class Node
{
 public:
      int val;
      Node *left;
      Node *right;
      Node(int val)
      {
           this->val = val;
           this->left = NULL;
           this->right = NULL;
      }
};
void parallel_dfs(Node *t)
```

```
{
      if(t = NULL)
            return;
      #pragma omp parallel sections
      {
            #pragma omp section
            {
                  parallel_dfs(t->left);
   std::cout<<t->val<<" ";
            }
            #pragma omp section
            {
                  parallel_dfs(t->right);
            }
      }
}
int main()
{
      int n = 10000;
      int arr[n];
      for(int i=0;i<n;i++)</pre>
            arr[i] = rand()% 100;
      Node *root = NULL;
```

```
if(root = NULL)
{
      Node *node = new Node(arr[0]);
      root = node;
}
for(int i=1;i<n;i++)
{
      Node *n = new Node(arr[i]);
      Node *temp = root;
      while(temp != NULL)
      {using namespace std::chrono;
            if(temp->val > arr[i])
            {
                  if(temp->left != NULL)
                        temp = temp->left;
                  else
                  {
                        temp->left = n;
                        break;
                  }
           }
            else
```

```
{
                        if(temp->right != NULL)
                              temp = temp->right;
                        else
                        {
                              temp->right = n;
                              break;
                        }
                  }
           }
     }
     time_point<system_clock> start,end;
      start = system_clock::now();
      parallel_dfs(root);
      end=system_clock::now();
  cout<<endl;
  duration<double> t = end-start;
      cout<<"The time required for parallel is :- "<<t.count()<<"ms"<<endl;</pre>
  return 0;
Output:
```

}

