Assignment No: 4

**Title: -**Parallel Search Algorithm-

Design and implement parallel algorithm utilizing all resources available. for

• Binary Search for Sorted Array

• Depth-First Search ( tree or an undirected graph ) OR

• Breadth-First Search ( tree or an undirected graph) OR

• Best-First Search that ( traversal of graph to reach a target in the shortest possible path)

**Binary Search:**

**Program:**

#include<iostream>

#include<stdlib.h>

#include<omp.h>

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

{

low1=low;

high1=mid;

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;

}

cout<<"here2";

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];

cout<<"\n enter elements=>";

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

{

cin>>a[i];

}

cout<<"\n enter key to find=>";

cin>>key;

loc=binary(a,0,n-1,key);

if(loc==-1)

cout<<"\n Key not found.";

else

cout<<"\n Key found at position=>"<<loc+1;

return 0;

}

**Output :**

