Binary Search
Algorithm Assignment

Dushyant Kumar Singh
Roll No. – 23

MCA 2nd Sem

1. If the statements {return BSearch(arr, mid + 1, high, key)} and {return BSearch(arr, low, mid-1, key)} are replaced by {return BSearch(arr, mid , high, key)} and {return BSearch(arr, low, mid, key)} respectively.
   Then the code will go in an infinite loop while searching for the rightmost element of the array or vector. For all other elements it will be able to search the correct index.

	0	1	2	3	4	5
Ex -	2	3	5	6	7	13

   While searching for 13 in the above example, the low and high will become 4 and 5 respectively in the last iteration. But from here every time the mid will be calculated as (4+5)/2 = 4.5 => Mid = 4. So, it will be an infinite loop.

2.

```
#include<iostream>
using namespace std;

//Code to find the position from where the rotation starts.
int bsearch_rotate_pos(int a[],int low, int high){
    if(low>high)
        return -1;
    int mid;
    mid = (low + high) / 2;
    if(a[mid] < a[mid-1])
        return mid-1;
    else if(a[mid] > a[mid+1])
        return mid;
    else if(a[mid] > a[high])
        return bsearch_rotate_pos(a,mid+1,high);
    else if(a[mid] < a[low])
        return bsearch_rotate_pos(a,low, mid-1);

    return 0;
}

//binary search function
int bsearch(int a[],int low, int high, int key){
    if(low>high)
        return -1;
    int mid = (low+high)/2;
    if(a[mid] == key){
        return mid;
    }
    else if(key < a[mid])
```

```
        return bsearch(a,low,mid-1,key);
    else if(key > a[mid])
        return bsearch(a,mid+1, high, key);

    return 0;
}




// in the function we are first find the position of rotation
and then calling the binary search on the two sorted arrays.
One array is from low till the position of rotation and other
array is from the position of rotation + 1 till high.
int driver(int a[],int low,int high, int key){
    int rpos = bsearch_rotate_pos(a,low,high);
    cout<<rpos<<"\n";
    int in = bsearch(a,low,rpos,key);
    if(in == -1)
        in = bsearch(a,rpos+1,high,key);

    return in;
}

int main(){
    int a[]={2,4,5,8,10,11,-2,0,1};
    cout<<driver(a,0,8,11);
    return 0;
}
```

3. The condition a[mid] == mid will help to find the required output where a[k] = k.
   It is same as the normal binary search. Only in place of key now we have mid itself to find the required element.

```
int bsearch(int a[],int low, int high){
   if(low>high)
      return -1;
   int mid = (low+high)/2;
   if(a[mid] == mid){
      return mid;
   }
   else if(mid < a[mid])
      return bsearch(a,low,mid-1);
   else if(mid > a[mid])
      return bsearch(a,mid+1, high);

   return 0;
}
int main(){

   int a[]={-1,0,3,4,5,6,7,8};
   cout<<bsearch(a,0,7);
   return 0;
}
```