1. **Binary Search (ascending order)**
2. *public static void* main(String[] args) {  
    *int*[] arr = {90,75,18,12,6,4,3,1};  
    *int* target= 75;  
    *int* out =*Binary\_Search*(arr,0, arr.length-1,target);  
    System.***out***.println(out);  
    }  
     
      
    *public static int* Binary\_Search(*int*[] arr,*int* s,*int* e,*int* target){  
    *if*(s>e){  
    *return* -1;  
    }  
    *int* mid= s + (s+e)/2;  
    *if*(arr[mid] == target){  
    *return* mid;  
    }  
    *if*(target>arr[mid]) {  
    *return Binary\_Search*(arr, s, mid, target);  
    }  
    *return Binary\_Search*(arr,mid+1,e,target);  
     
    }

**2 . Valid Perfect Square**

*public static void* main(String[] args) {  
  
 *int* target = 121;  
 *int* out = *Binary\_Search*( 0, target, target);  
 System.***out***.println(out);  
}  
  
  
*public static int* Binary\_Search( *int* s, *int* e, *int* target) {  
 *if* (s >= e) {  
 *return* -1;  
 }  
 *if*(e<=0){  
 *return* -1;  
 }  
 *int* mid = s+(e-s) / 2;  
 *if* (mid\*mid == target) {  
 *return* mid;  
 }  
 *if* (mid\*mid>target) {  
 *return Binary\_Search*( s, mid, target);  
 }  
 *return Binary\_Search*( mid + 1, e, target);  
  
}

1. **Guess number**
2. *public static void* main(String[] args) {  
    *int* res=*Binary\_Search*(0,10,4);  
    System.***out***.println(res);  
    }  
     
    *public static int* Binary\_Search(*int* s,*int* e,*int* target){  
    *if*(s>e){  
    *return* -1;  
    }  
    *int* mid=s+(e-s)/2;  
    *if*(*guess*(mid,target)==0){  
    *return* mid;  
    }  
    *if*(*guess*(mid,target)<0){  
    *return Binary\_Search*(s,mid,target);  
    }  
    *return Binary\_Search*(mid+1,e,target);  
    }  
    *public static int* guess(*int* num,*int* target){  
    *if*(num==target){  
    *return* 0;  
    }  
    *if*(target<num){  
    *return* -1;  
    }  
     
    *return* 1; }

**4.Find peak element**

*public static void* main(String[] args) {  
 *int*[] arr= {1,2,3,4,5};  
*// int[] arr= {7,8,9,6,4};  
// int[] arr= {2,3,4,1,-4};  
// int[] arr= {6,4,3,2,1};  
 int* res= *Search\_peak*(arr,0, arr.length-1);  
 System.***out***.println(res);  
 }  
  
 *public static int* Search\_peak(*int*[] arr,*int* s,*int* e){  
  
 *if*(s== arr.length-1){  
 *return* arr.length-1;  
 }  
 *int* mid=s+(e-s)/2;  
 *if*(mid==0){  
 *return* mid;  
 }  
 *if*(arr[mid-1]<arr[mid]&&arr[mid]>arr[mid+1]){  
 *return* mid;  
 }  
 *if*(arr[mid+1]>arr[mid]){  
 *return Search\_peak*(arr,mid+1,e);  
 }  
 *return Search\_peak*(arr,s,mid);  
 }

1. **Square Root**
2. *public static void* main(String[] args) {  
   *// System.out.println(Square\_Root(0,6,6));* System.***out***.println(*Square\_Root*(0,10,10));  
    }  
     
    *public static int* Square\_Root(*int* s,*int* e,*int* target){  
    *if*(s==e&&e\*e>target){  
    *return* e-1;  
    }  
    *int* mid=s+(e-s)/2;  
    *if*(mid\*mid==target){  
    *return* mid;  
    }  
    *if*(mid\*mid>target){  
    *return Square\_Root*(s,mid,target);  
    }  
    *return Square\_Root*(mid+1,e,target);  
    }

**Recursion**

**6.Power of Two**

*public static void* main(String[] args) {  
 System.***out***.println(*Power\_of\_Two*(129));  
}  
  
*public static boolean* Power\_of\_Two(*int* num){  
 *if*(num==2){  
 *return true*;  
 }  
 *if*(num%2!=0 || num ==1){  
 *return false*;  
 }  
 *return Power\_of\_Two*(num/2);  
}

1. **Find power of a number**
2. *public static void* main(String[] args) {  
    System.***out***.println(*Power\_of\_number*(9,3));  
    System.***out***.println(*Power\_of\_number*(5,5));  
   }  
     
   *public static int* Power\_of\_number(*int* b,*int* p){  
    *if*(p==0){  
    *return* 1;  
    }  
    *return* b\**Power\_of\_number*(b,p-1);  
   }

**8.Ninja and sorted array**

*public static void* main(String[] args) {  
 *int*[] arr1={3,6,9,11,19,0,0,0,0,0};  
 *int*[] arr2={4,10,12,15,20};  
 arr1=*mergeArray*(arr1,0,arr2,0);  
 System.***out***.println(Arrays.*toString*(arr1));  
}  
*public static int*[] mergeArray(*int*[] arr1,*int* p1,*int*[] arr2,*int* p2){  
 *if*(arr1[p1]==0){  
 *sort\_arr*(arr2);  
 *for*(*int* i=0;i<arr2.length;i++){  
 arr1[p1++]=arr2[i];  
 }  
 *return* arr1;  
 }  
 *if*(arr1[p1]<arr2[p2]){  
 p1++;  
 *return mergeArray*(arr1,p1,arr2,p2);  
 } *else* {  
 *if*(arr2[p2]>arr2[p2+1]){  
 arr2=*sort\_arr*(arr2);  
 }  
 *int* temp = arr1[p1];  
 arr1[p1] = arr2[p2];  
 arr2[p2] = temp;  
 p1++;  
  
 *return mergeArray*(arr1, p1, arr2, p2);  
 }  
  
  
}  
  
*private static int*[] sort\_arr(*int*[] arr2) {  
 *for*(*int* i=1;i< arr2.length;i++){  
 *int* temp=arr2[i];  
 *int* j=i-1;  
 *while*(j>=0&&arr2[j]>temp){  
 arr2[j+1]=arr2[j];  
 j--;  
 }  
 arr2[j+1]=temp;  
 }  
 *return* arr2;  
}

1. **Reverse a string**

*public static void* main(String[] args) {  
 String str="abcde";  
 System.***out***.println(str);  
 *char*[] str\_arr=*new char*[str.length()];  
 str\_arr=str.toCharArray();  
 str\_arr=*reverse\_string*(0,str\_arr);  
 str=Arrays.*toString*(str\_arr);  
 System.***out***.println(str);  
  
}  
  
*private static char*[] reverse\_string( *int* p,*char*[] str\_arr) {  
 *if*(p>str\_arr.length/2){  
 *return* str\_arr;  
 }  
 *char* temp=str\_arr[p];  
 str\_arr[p]=str\_arr[str\_arr.length-1-p];  
 str\_arr[str\_arr.length-1-p]=temp;  
 *return reverse\_string*(++p,str\_arr);  
}