#양수/음수/제로 판결

```
main() {
  int inputValue;
  char* decision;
  scanf( “%d” , &inputValue); // 입력받음
  if(inputValue>0) // 양수이면
    decision="양수";
  else if(inputValue<0) // 음수이면
    decision="음수";
  else decision="제로"; // 양수, 음수가 아니면
  printf( “%s” , decision);
}
```

#홀수 출력하기

```
main() {
  int limit;
  scanf( “%d” , &limit); //입력을 받음
  for(i=1; i<=limit; i=i+2) //순환하며 출력
  {
    printf( “%d” , i);
  }
}
```

#홀수 더하기 짝수 빼기

```
main() {
  int sum=0, flag=0;;
  for(i=1; i=100; i=i + 1)
  {
    if(flag==1)
    {
      sum=sum-I;
      flag=0
    } else {
      sum=sum+I;
      flag=1;
    }
  }
  printf( “%d” , sum);
}
```

#50개 배열 평균 구하기

```
main() {
  int student[50], total=0, average;
  for(i=1; i<=49; i=i + 1)
  {
    scanf( “%d” , &student[i]);
    total=total+student[i];
  }
  average=total/50;
  printf( “%d” , average);
}
```

```
#학번과 점수 출력

main() {
    int score[2][30], num;
    scanf( “%d” , &num);
    for(i=1; i<30; i=i + 1)
    {
        if(score[0][i]==num
        {
            printf( “%d” , score[1][i];
            break;
        }
    }
}


class Node : # 연결 리스트를 구성하는 단위 데이터의 모습은 data+다음 데이터
    def __init__(self, data, next=None) :
        self.data = data
        self.next=next

def init() : # 연결 리스트를 생성한다. node 1~node 4 그리고 연결 포인터 구성
    global node1
    node1 = Node(1)
    node2 = Node(2)
    node3 = Node(3)
    node4 = Node(4)
    node1.next = node2
    node2.next = node3
    node3.next = node4

def delete(del_data) : # 구성된 리스트에서 데이터를 지우고, 나머지를 연결한다.
    global node1
    pre_node = node1
    next_node = pre_node.next

    if pre_node.data == del_data :
        node1 = next_node
        del pre_node
        return
    while next_node :
        if next_node.data == del_data:
            pre_node.next = next_node.next
            del next_node
            break
        pre_node = next_node
        next_node = next_node.next

def insert(ins_data) : # 연결 리스트에 데이터를 추가한다.
    global node1
    new_node = Node(ins_data)
    new_node.next=node1
    node1 = new_node

def print_list() : # 연결 리스트의 데이터를 출력한다.
    global node1
    node = node1
    while node:
        print (node.data)
        node=node.next
    print()

def LinkedList() :
    init();
    delete(2)
    insert("9")
    print_list()

LinkedList()
```

```
public class Tree {
    TreeNode topNode=null;
    public void add(int key,Object value) {

    if(topNode==null)
        topNode=new TreeNode(key, value);
    else
        topNode.add(key, value);
    }
    public Object get(int key) {
        return topNode==null ? null:topNode.find(key);
    }
}


def Queue() :
    queue = []
    queue.append(1)
    queue.append(2)
    queue.append(3)
    queue.append(4)
    queue.append(5)
    print(queue)
    while queue :
        print("Get Value :", queue.pop(0))
Queue()


public class TreeNode {

private int itsKey;
private Object itsValue;
private TreeNode nodes[]=new TreeNode[2];

public TreeNode(int key, Object value) {
    itsKey=key;
    itsValue=value;
    System.out.println("start TreMapNode");
}

public Object find(int key) {
    if(key == itsKey ) {
        return itsValue;
    }
    return findSub(selectSubNode(key), key);
}

private int selectSubNode(int key) {
    return (key <itsKey) ? 0 : 1;
}

private Object findSub(int node, int key) {
    return nodes[node]==null ?
        null:nodes[node].find(key);
}

public void add(int key, Object value) {
    if(key == itsKey)
        itsValue=value;
    else
        SubNode(selectSubNode(key), key, value);
}

private void SubNode(int node, int key, Object value) {
    if(nodes[node]==null)
        nodes[node]=new TreeNode(key, value);
    else
        nodes[node].add(key, value);
    }
}
```

```
def radix(order) :
    is_sorted = False
    position = 1

    while not is_sorted :
        is_sorted = True
        queue_list = [list() for _ in range(10)]

        for num in order :
            digit_number = (int) (num/position) % 10
            queue_list[digit_number].append(num)
            if is_sorted and digit_number >0 :
                is_sorted = False
        index=
        for numbers in queue_list :
            for num in numbers:
                order[index] =num
                index += 1
        position *=10

x = [5,2,8,6,1,9,3,7]
radix(x)


def change(x, i, j) :
    x[i], x[j] = x[j], x[i]

def selectionSort(x) :
    for size in reversed(range(len(x))) :
        max_i = 0
        for i in range(1, 1+size) :
            if x[i] > x[max_i] :
                max_i = i
        change(x, max_i, size)

x = [5,2,8,6,1,9,3,7]
selectionSort(x)


def mergeSort(x) :
    if len(x) > 1 :
        mid = len(x) // 2
        colx, rowx = x[:mid], x[mid:]
        mergeSort(colx)
        mergeSort(rowx)

    coli, rowi, i = 0,0,0

    while coli < len(colx) and rowi < len(rowx) :
        if colx[coli] < rowx[rowi] :
            x[i] = colx[coli]
            coli += 1
        else :
            x[i] =rowx[rowi]
            rowi +=1
        i +=1
    x[i:] =colx[coli:] if coli != len(colx) else
    rowx[rowi :]

x = [5,2,8,6,1,9,3,7]
mergeSort(x)
```

```
def insertSort(x) :
    for size in range(1, len(x)) :
        val= x[size]
        i = size
    while i>0 and x[i-1] >val :
        x[i] = x[i-1]
        i -= 1
    x[i] = val

x = [5,2,8,6,1,9,3,7]
insertSort(x)


def Between(x, start, ranges) :
    for target in range(start+ranges, len(x), ranges) :
        val = x[target]
        i = target
        while i > start :
        if x[i - ranges] > val :
            x[i] = x[i - ranges]
        else :
            break
        i -= ranges
    x[i] =val

def shellSort(x) :
    ranges=len(x)//2
    while ranges >0 :
        for start in range(ranges) :
            Between(x, start, range)
        ranges= ranges//2
x = [5,2,8,6,1,9,3,7]

shellSort(x)


def change(x, i, j) :
    x[i], x[j] = x[j], x[i]

def Select(x, l, r) :
    select_val = x[l]
    select_idx = l
    while l<= r :
        while l <= r and x[l] <= select_val:
            l += 1
        while l <= r and x[r] >= select_val:
            r -=1
        if l <= r:
            change(x, l, r)
            l =+ 1
            r -= 1
    change(x, select_idx, r)
    return r

def quickSort(x, pivotMethod = Select) :
        def Qsort(x, first, last) :
            if first < last :
                splitP = pivotMethod(x, first, last)
                Qsort(x, first, splitP-1)
                Qsort(x, splitP+1, last)
    Qsort(x, 0, len(x)-1)


x = [5,2,8,6,1,9,3,7]
quickSort(x)
```

```
def sequentialSearch(array, value) :
    for i in range(len(array)) :
        if array[i] == value:
            return i
    return False

x = [5,2,8,6,1,9,3,7]
i = sequentialSearch(x, 3)
print(i)
```

```
public class LinearList {
    public static void main(String args[]) {
        int sale[] = new int [] {12, 45, 67, 43, 56, 98}; // 리스트 구조의 선언
        for(int i=0; i<6; i++) // 리스트 구조의 출력
            System.out.printf("The sales result = %d %n", sale[i]);
    }
}
```

```
public class Link {
    public int data1;
    public double data2;
    public Link nextLink;

    public Link(int d1, double d2) {
        data1 = d1;
        data2 = d2;
    }

    public void printLink() {
        System.out.print("{" + data1 + ", " + data2 + "} ");
    }
}
```

```
class LinkListTest {
    public static void main(String[] args) {
        LinkList list = new LinkList();

        list.insert(1, 100);
        list.insert(2, 200);
        list.insert(3, 300);
        list.insert(4, 400);
        list.insert(5, 500);
        list.printList();

        while(!list.isEmpty()) {
            Link deletedLink = list.delete();
            System.out.print("deleted: ");
            deletedLink.printLink();
            System.out.println("");
        }
        list.printList();
    }
}
```

```
import java.util.*;

public class Stack {
    public static void main(String[] args) {
        java.util.Stack<Integer> s = new java.util.Stack<Integer>();
        System.out.println("Stack created :");
        for(int i =0; i <10;i++) // 0~9의 수로 스택을 구성한다
            s.push(new Integer(i));
        System.out.println("1pop:" + s.pop()); // 스택의 값은 뺀다
        System.out.println("2pop:" + s.pop());
        System.out.println("3pop:" + s.pop());
        System.out.println("4pop:" + s.pop());
        System.out.println("stack top :" + s.peek()); // 현재 스택의 위치를 보여준다
    }
}
```

```
def Queue() :
    queue = []
    queue.append(1)
    queue.append(2)
    queue.append(3)
    queue.append(4)
    queue.append(5)
    print(queue)
    while queue :
        print("Get Value :", queue.pop(0))
Queue()
```

```
public class Main {
    public static void main(String[] args) {
        Tree TM=new Tree();
        TM.add(10, "cho");
        TM.add(20, "KIM");
        TM.add(30, "minho");
        TM.add(40, "JYeon");
        System.out.println("Data Search and Get...");
        Object Temp=TM.get(20);
        System.out.println(Temp);
        System.out.println("Data Search and Get...");
        Object Temp2=TM.get(40);
        System.out.println(Temp2);
    }
}
```

```
public class Tree {
    TreeNode topNode=null;
    public void add(int key,Object value) {
        if(topNode==null)
            topNode=new TreeNode(key, value);
        else
            topNode.add(key, value);
    }
    public Object get(int key) {
        return topNode==null ? null:topNode.find(key);
    }
}
```

```
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

public class QuickSort {
    public static void main(String[] args) {
        Integer[] array = { 30, 50, 7, 40, 88, 15, 44, 55, 22, 33, 77, 99, 11, 66, 1, 85 };
        ArrayList<Integer> aList = new ArrayList<Integer>();
        aList.addAll(Arrays.asList(array));
        aList = (ArrayList<Integer>) quicksort(aList);
        out.print(aList.toString());
    }
    public static <T extends Comparable<? super T>> List<T> quicksort(List<T> list) {
        if (list.size() <= 1) return list;
        int pivot = list.size() / 2;
        List<T> a = new ArrayList<T>(); // lesser
        List<T> b = new ArrayList<T>(); // greater
        int c = 0; // same
        for (T number : list) {
            if (list.get(pivot).compareTo(number) < 0)
                b.add(number);
            else if (list.get(pivot).compareTo(number) > 0)
                a.add(number);
            else
                c++;
        }
        a = quicksort(a);
        for (int i = 0; i < c; i++)
            a.add(list.get(pivot));
        b = quicksort(b);
        List<T> sorted = new ArrayList<T>();
        sorted.addAll(a);
        sorted.addAll(b);
        return sorted;
    }
}


import static java.lang.System.out;
import java.util.Arrays;

public class SelectionSort {
    public static void main(String[] args) {
        int [] array = {12, 326, 127, 467, 110, 58};
        int size=6;
        array=SelectionSort(array, size);
        out.print(Arrays.toString(array));
    }
    public static int [] SelectionSort(int arr[], int MAX) {
        int i, j;
        int min, temp;
        for(i=0; i<MAX-1; i++) {
            min = i;
            for(j=i+1; j<MAX; j++) {
                if(arr[j] < arr[min]) min = j;
            }
            temp = arr[i];
            arr[i] = arr[min];
            arr[min] = temp;
        }
        return arr;
    }
}
```

```
import static java.lang.System.out;
import java.util.Arrays;

public class ExchangeSort {
    public static void main(String[] args) {
        int [] array = {12, 326, 127, 467, 110, 58};
        array=ExchangeSort(array);
        out.print(Arrays.toString(array));
    }
    public static int [] ExchangeSort(int arr[]) {
        int i, j;
        int temp;
        int numLength=6;
        for(i=0; i<numLength-1; i++) {
            for( j = (i+1); j <numLength; j++){
                if(arr[i] >arr[j]) {
                    temp= arr[i];
                    arr[i] = arr[j];
                    arr[j] = temp;
                }
            }
        }
        return arr;
    }
}


public class InsertSort {
    public static void main (String[] args) {
        int[] a = {12, 326, 127, 467, 110, 58};
        for(int j = 1; j < a.length; j++) {
            int key = a[j];
            int i = j-1;
            while(i >= 0 && a[i] > key) {
                a[i+1] = a[i];
                i = i - 1;
            }
            a[i+1]=key;
        }
        for(int i = 0; i < a.length; i++) {
            System.out.print(a[i] + " ");
        }
    }
}


public class ShellSort {
    public static void intervalSort(int arr[], int begin, int end, int interval) {
        int item=0;
        int j=0;
        for(int i=begin+interval; i<=end; i=i+interval) {
            item=arr[i];
            for(j=i-interval; j >=begin && item < arr[j]; j -=interval)
                arr[j+interval]=item;
        }
    }
    public static void shellSort(int arr[]) {
        int interval=0;
        int t=1;
        int arrSize=arr.length;
        interval=arrSize/2;
        while(interval >= 1) {
            for(int i
```

```
import static java.lang.System.out;
import java.util.Arrays;

public class RadixSort {
    public static void main(String[] args) {
        int [] array = {12, 326, 127, 467, 110, 58};
        array=rSort(array);
        out.print(Arrays.toString(array));
        // 결과는 12, 58, 110, 127, 326, 467로 나온다
    }
    public static int [] rSort(int [] array) {
            for(int shift=Integer.SIZE - 1; shift > -1 ; shift--) {
                int [] tmp=new int[array.length];
                int j=0;
                for ( int i =0; i <array.length; i++) {
                    boolean move=array[i] << shift >=0;
                    if(shift == 0 ? ! move:move) {
                        tmp[j] = array[i];
                        j++;
                    } else {
                    array[i-j] = array[i];
                }
            }
            for(int i = j; i <tmp.length;i++) {
                tmp[i] = array[i-j];
            }
            array =tmp;
        }
        return array;
    }
}


import static java.lang.System.out;
import java.util.Arrays;

public class SelectionSort {
    public static void main(String[] args) {
        int [] array = {12, 326, 127, 467, 110, 58};
        int size=6;
        array=SelectionSort(array, size);
        out.print(Arrays.toString(array));
    }
    public static int [] SelectionSort(int arr[], int MAX) {
        int i, j;
        int min, temp;
        for(i=0; i<MAX-1; i++) {
            min = i;
            for(j=i+1; j<MAX; j++) {
                if(arr[j] < arr[min]) min = j;
            }
            temp = arr[i];
            arr[i] = arr[min];
            arr[min] = temp;
        }
        return arr;
    }
}
```

```
import static java.lang.System.out;
import java.util.Arrays;

public class ExchangeSort {
    public static void main(String[] args) {
        int [] array = {12, 326, 127, 467, 110, 58};

        array=ExchangeSort(array);
        out.print(Arrays.toString(array));
    }

    public static int [] ExchangeSort(int arr[]) {
        int i, j;
        int temp;
        int numLength=6;
        for(i=0; i<numLength-1; i++) {
            for( j = (i+1); j <numLength; j++){
                if(arr[i] >arr[j]) {
                    temp= arr[i];
                    arr[i] = arr[j];
                    arr[j] = temp;
                }
            }
        }
        return arr;
    }
}



public class InsertSort {
    public static void main (String[] args) {
        int[] a = {12, 326, 127, 467, 110, 58};
        for(int j = 1; j < a.length; j++) {
            int key = a[j];
            int i = j-1;
            while(i >= 0 && a[i] > key) {
                a[i+1] = a[i];
                i = i - 1;
            }
            a[i+1]=key;
        }
        for(int i = 0; i < a.length; i++) {
            System.out.print(a[i] + " ");
        }
    }
}


public class ShellMain {
    public static void main(String[] args) {
        int [] arr = {23, 12, 45, 3, 18, 32, 49};
        ShellSort.shellSort(arr);
    }
}
```

```
public class ShellSort {
    public static void intervalSort(int arr[], int begin, int end, int interval) {
        int item=0;
        int j=0;
        for(int i=begin+interval; i<=end; i=i+interval) {
            item=arr[i];
            for(j=i-interval; j >=begin && item < arr[j]; j -=interval)
                arr[j+interval]=item;
        }
    }
    public static void shellSort(int arr[]) {
        int interval=0;
        int t=1;
        int arrSize=arr.length;
        interval=arrSize/2;
        while(interval >= 1) {
            for(int i =0; i<interval; i++)
                intervalSort(arr, i, arrSize-1, interval);
            System.out.println("셸 정렬 "+ t++ +" 단계 : interval =>" +interval);
            System.out.println(arr);
            interval/=2;
        }
    }
}


import java.util.Scanner;

public class SequentialSearch {
    public static void main(String[] args) {
        int[] dataArray = { 4, 21, 2, 10, 11, 16 };
        System.out.println("검색할 데이터를 입력하세요");
        Scanner scan = new Scanner(System.in);
        int search = Integer.parseInt(scan.nextLine().trim()); // 데이터를 입력받습니다.
        int result = sequentialSearch(dataArray, search);
        if(result == -1)
            System.out.println("데이터를 찾지 못하였습니다");
        else
            System.out.println("데이터의 위치는 " + result + "번째 입니다.");
    }
    public static int sequentialSearch(int[] arr, int search) {
        for (int i = 0; i < arr.length; i++) { // 순서대로 비교하기 위해 배열의 크기만큼 반복합니다.
            if (arr[i] == search) // 비교할 데이터가 배열에 있으면 배열의 위치를 return하고, 없다면 -1을 return 합니다.
                return i;
        }
        return -1;
    }
}
```

```java
import java.util.Scanner;

public class BinarySearch {
    public static void main(String[] args) {
        int[] dataArray = { 1, 2, 3, 4, 5, 6, 7, 8, 9 };
        System.out.println("검색할 데이터를 입력하세요");
        Scanner scan = new Scanner(System.in);
        int search = Integer.parseInt(scan.nextLine().trim()); // 데이터를 입력받습니다.
        binarySearch(dataArray, search); // 이진 검색 모듈의 수행
        return;
    }
    public static void binarySearch(int arr[], int iKey) {
        int mid;
        int left = 0;
        int right = arr.length - 1;
        while (right >= left) {
            mid = (right + left) / 2;
            if (iKey == arr[mid]) {
                System.out.println(iKey + " is in the array with index value: " + mid);
                break;
            }
            if (iKey < arr[mid]) {
                right = mid - 1;
            } else {
                left = mid + 1;
            }
        }
    }
}


public class StringAlgorithm {
    public static void main(String[] args) {
        String str = "abcdabcdfgab";
        String sStr = "ab";
        int orStrLen = str.length(); // 오리지널 스트링
        int sStrLen = sStr.length(); // 찾으려는 단어
        char[] searchString= sStr.toCharArray(); // 찾을 단어를 character형 배열로 변환
        char[] allString = str.toCharArray(); // 전체 문장을 character형 배열로 변환
        int start; // 전체의 문장에서 찾으려는 단어를 빼서 시작 인덱스를 구한다.
        int i;
        int count = 0; // 문장에서 찾으려는 개수
        for(start=0; start < orStrLen ; start++) { // 전체 문장의 처음부터 시작
            for( i=0; i<sStrLen; i++ ) { // 찾으려는 단어를 대입
                if( allString[start] == searchString[i] ) { // 각 단어의 첫 번째 단어의 비교
                    if( allString[start+1] == searchString[i + 1] ) { // 두 번째 단어의 비교
                        count++; // 두 개의 단어가 같으면 하나 증가
                    }
                } else {
                    break;
                }
            }
        }
        System.out.println("찾아진 단어의 개수 = " + count);
    }
}
```

```
public class KMP {
    private final int R; // the radix
    private int[][] dfa; // the KMP automoton
    private String pat; // or the pattern string
    public KMP(String pat) {
        this.R = 256;
        this.pat = pat;
        int m = pat.length();
        dfa = new int[R][m];
        dfa[pat.charAt(0)][0] = 1;
        for (int x = 0, j = 1; j < m; j++) {
            for (int c = 0; c < R; c++)
                dfa[c][j] = dfa[c][x]; // Copy mismatch cases.
            dfa[pat.charAt(j)][j] = j+1; // Set match case.
            x = dfa[pat.charAt(j)][x]; // Update restart state.
        }
    }
    public int search(String txt) {
        int m = pat.length();
        int n = txt.length();
        int i, j;
        for (i = 0, j = 0; i < n && j < m; i++) {
            j = dfa[txt.charAt(i)][j];
        }
        if (j == m) return i - m; // found
        return n; // not found
    }
    public static void main(String[] args) {
```

```
public class BMAlgorithm
{
```