Q1.

1.Use an array of classes, where each element of the array is a class that has both “key” and “value” data.

>O(n) get O(n) put O(n) remove

2.Use a linkedlist of classes, where each element of the array is a class that has both “key” and “value” data.

>O(n) get O(n) put O(n) remove

Q2.

Public static void sortByScore (ArrayList<Student> students){

Collections.sort (students, new studentComp());

}

class studentComp implements Comparator<Student>{

public int compare (student a, student b){

if (a.getScore() != b.getScore()){

return b.getScore() – a.getScore();

}

else{

return a.getName().compareTo (b.getName());

}

}

Q3.

A.

Public class stack {

private int[] a;

private int n;

public stack () {

a = new int[2];

n = 0;

}

public boolean isEmpty() {

return n==0;

}

public int top() {

return a[n-1];

}

public void push (int val) {

if(n==a.length){

a = Arrays.copyOf (a, 2\*n);

}

n++;

a[n-1] = val;

}

public int pop () {

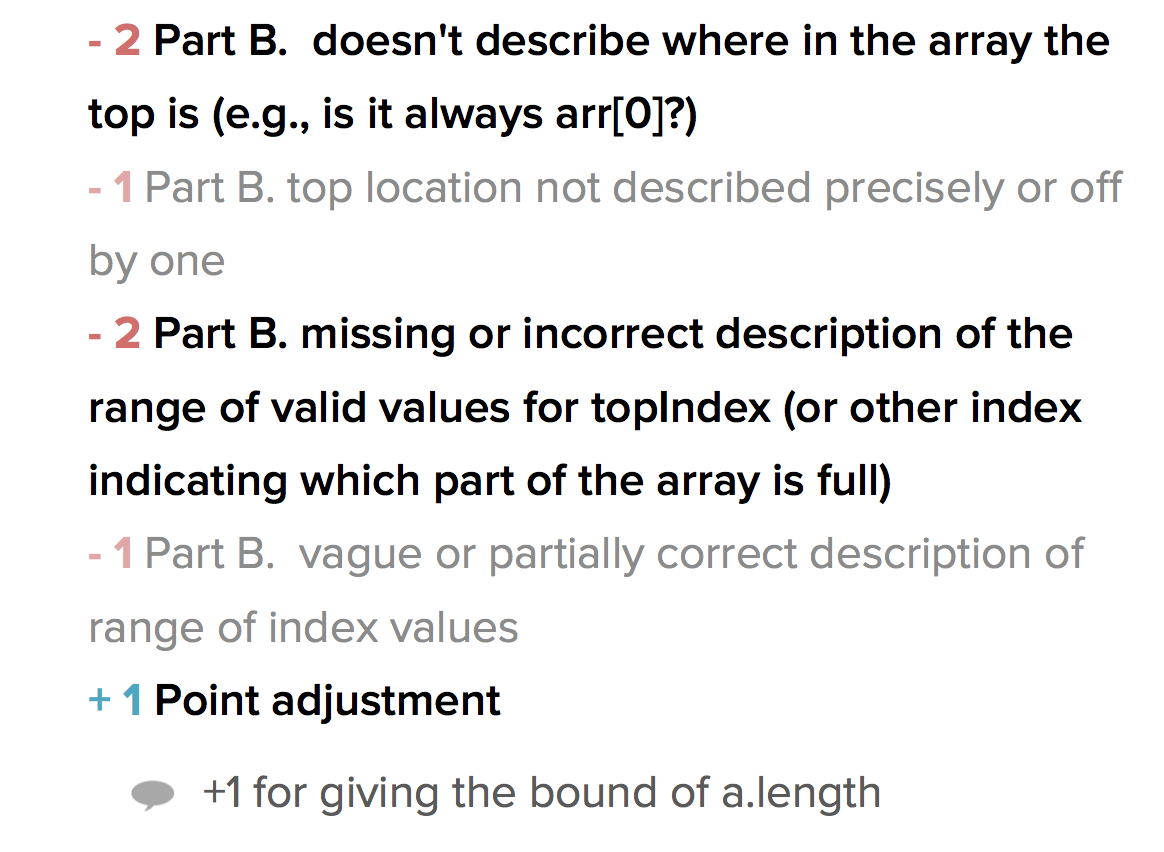
int temp = a[n-1];

a[n-1] =null;

n--;

return temp;

}



B. (這題不完全正確)

n>=0, n=0 when stack is empty

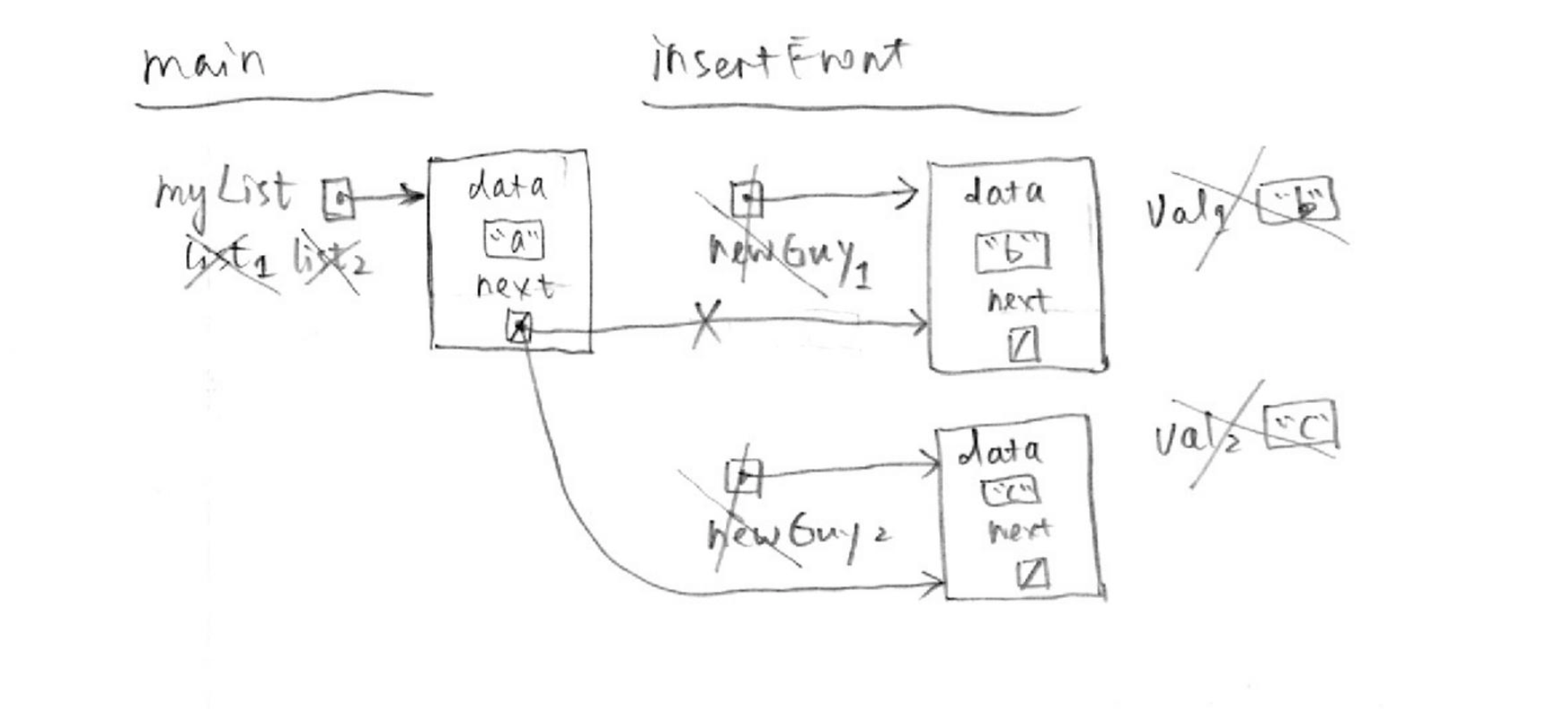
n<= a.length<2n

a[k] is null for n <=k<a.length

a[k] is not null for 0<=k<n

Q4.

A.



B.

>a d

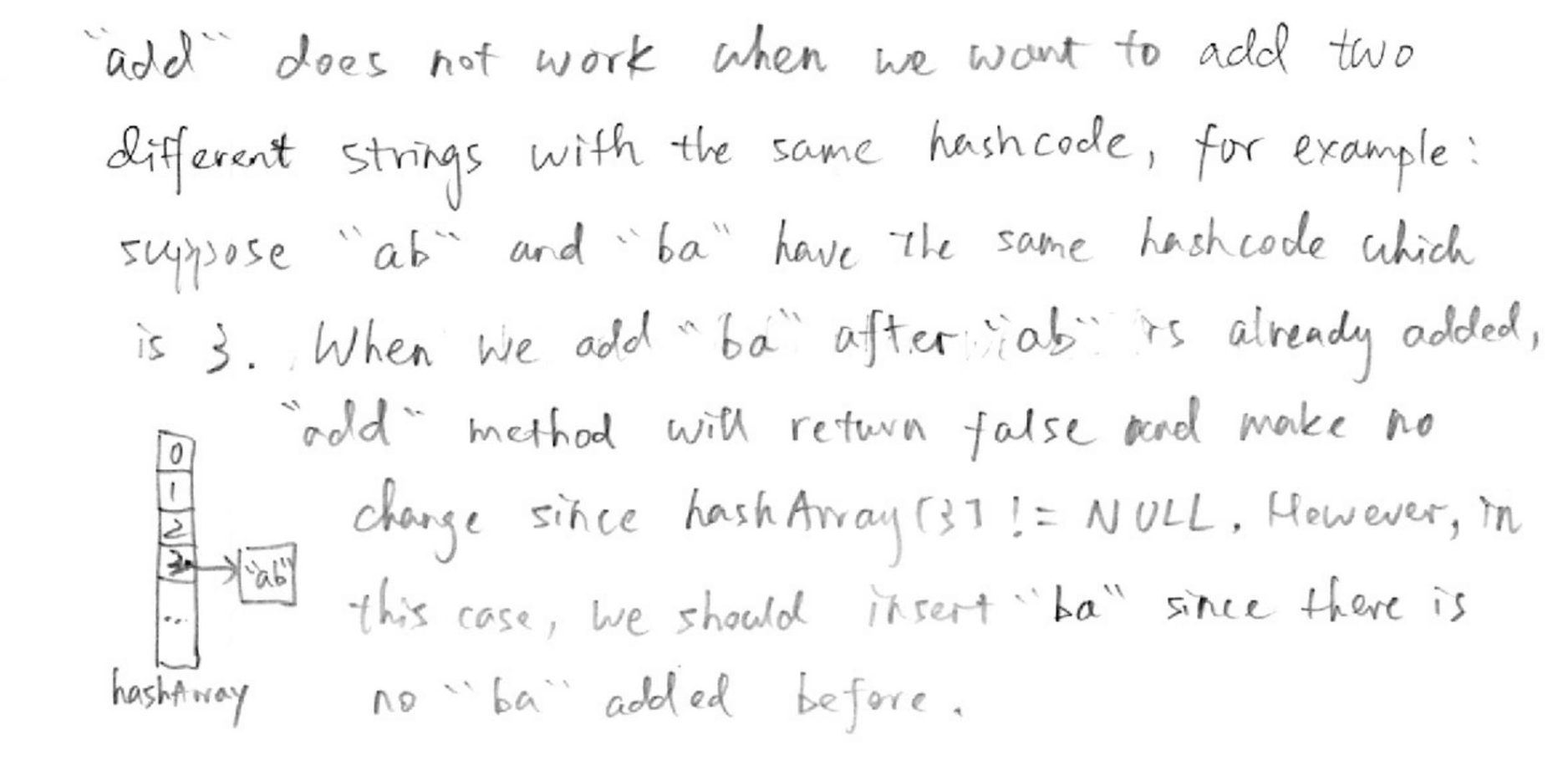
Q5.

void initCounts (int counts[], int size);

void readAndCompute (int counts[], int size);

void printHist (int counts[], int size);

Q6.



Q7.

void gut (ListType & list) {

ListType node = list;

if(node != NULL){

if(node->next != NULL){

while(node->next->next !=NULL){

ListType temp = node->next;

Node->next = node->next->next;

Delete temp;

}

}

}

}