#include <assert.h>

#include <ctype.h>

#include <limits.h>

#include <math.h>

#include <stdbool.h>

#include <stddef.h>

#include <stdint.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

char\* readline();

char\* ltrim(char\*);

char\* rtrim(char\*);

char\*\* split\_string(char\*);

int parse\_int(char\*);

/\*

 \* Complete the 'maxSubarray' function below.

 \*

 \* The function is expected to return an INTEGER\_ARRAY.

 \* The function accepts INTEGER\_ARRAY arr as parameter.

 \*/

/\*

 \* To return the integer array from the function, you should:

 \*     - Store the size of the array to be returned in the result\_count variable

 \*     - Allocate the array statically or dynamically

 \*

 \* For example,

 \* int\* return\_integer\_array\_using\_static\_allocation(int\* result\_count) {

 \*     \*result\_count = 5;

 \*

 \*     static int a[5] = {1, 2, 3, 4, 5};

 \*

 \*     return a;

 \* }

 \*

 \* int\* return\_integer\_array\_using\_dynamic\_allocation(int\* result\_count) {

 \*     \*result\_count = 5;

 \*

 \*     int \*a = malloc(5 \* sizeof(int));

 \*

 \*     for (int i = 0; i < 5; i++) {

 \*         \*(a + i) = i + 1;

 \*     }

 \*

 \*     return a;

 \* }

 \*

 \*/

 int max(int a,int b){

    return (a>b)?a:b;

 }

int\* maxSubarray(int arr\_count, int\* arr, int\* result\_count) {

\*result\_count=2;

int\* result=malloc(2\* sizeof(int));

int max\_end=arr[0];

int max\_so=arr[0];

for(int i=1;i<arr\_count;i++){

    max\_end=max(arr[i],max\_end+arr[i]);

    max\_so=max(max\_so,max\_end);

}

result[0]=max\_so;

int subseq\_sum=0;

int has\_pos=0;

int max\_ele=arr[0];

for(int i=0;i<arr\_count;i++){

    if(arr[i]>0){

        subseq\_sum+=arr[i];

        has\_pos=1;

    }

    if(arr[i]>max\_ele){

        max\_ele=arr[i];

    }

}

if(has\_pos)

result[1]=subseq\_sum;

else

 result[1]=max\_ele;

 return result;;

}

int main()

{

    FILE\* fptr = fopen(getenv("OUTPUT\_PATH"), "w");

    int t = parse\_int(ltrim(rtrim(readline())));

    for (int t\_itr = 0; t\_itr < t; t\_itr++) {

        int n = parse\_int(ltrim(rtrim(readline())));

        char\*\* arr\_temp = split\_string(rtrim(readline()));

        int\* arr = malloc(n \* sizeof(int));

        for (int i = 0; i < n; i++) {

            int arr\_item = parse\_int(\*(arr\_temp + i));

            \*(arr + i) = arr\_item;

        }

        int result\_count;

        int\* result = maxSubarray(n, arr, &result\_count);

        for (int i = 0; i < result\_count; i++) {

            fprintf(fptr, "%d", \*(result + i));

            if (i != result\_count - 1) {

                fprintf(fptr, " ");

            }

        }

        fprintf(fptr, "\n");

    }

    fclose(fptr);

    return 0;

}

char\* readline() {

    size\_t alloc\_length = 1024;

    size\_t data\_length = 0;

    char\* data = malloc(alloc\_length);

    while (true) {

        char\* cursor = data + data\_length;

        char\* line = fgets(cursor, alloc\_length - data\_length, stdin);

        if (!line) {

            break;

        }

        data\_length += strlen(cursor);

        if (data\_length < alloc\_length - 1 || data[data\_length - 1] == '\n') {

            break;

        }

        alloc\_length <<= 1;

        data = realloc(data, alloc\_length);

        if (!data) {

            data = '\0';

            break;

        }

    }

    if (data[data\_length - 1] == '\n') {

        data[data\_length - 1] = '\0';

        data = realloc(data, data\_length);

        if (!data) {

            data = '\0';

        }

    } else {

        data = realloc(data, data\_length + 1);

        if (!data) {

            data = '\0';

        } else {

            data[data\_length] = '\0';

        }

    }

    return data;

}

char\* ltrim(char\* str) {

    if (!str) {

        return '\0';

    }

    if (!\*str) {

        return str;

    }

    while (\*str != '\0' && isspace(\*str)) {

        str++;

    }

    return str;

}

char\* rtrim(char\* str) {

    if (!str) {

        return '\0';

    }

    if (!\*str) {

        return str;

    }

    char\* end = str + strlen(str) - 1;

    while (end >= str && isspace(\*end)) {

        end--;

    }

    \*(end + 1) = '\0';

    return str;

}

char\*\* split\_string(char\* str) {

    char\*\* splits = NULL;

    char\* token = strtok(str, " ");

    int spaces = 0;

    while (token) {

        splits = realloc(splits, sizeof(char\*) \* ++spaces);

        if (!splits) {

            return splits;

        }

        splits[spaces - 1] = token;

        token = strtok(NULL, " ");

    }

    return splits;

}

int parse\_int(char\* str) {

    char\* endptr;

    int value = strtol(str, &endptr, 10);

    if (endptr == str || \*endptr != '\0') {

        exit(EXIT\_FAILURE);

    }

    return value;

}