27ος ΠΑΝΕΛΛΗΝΙΟΣ ΔΙΑΓΩΝΙΣΜΟΣ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΕΝΔΕΙΚΤΙΚΕΣ ΛΥΣΕΙΣ Β΄ ΦΑΣΗΣ

ΛΥΚΕΙΟ
«Η μοιρασιά»

PASCAL

ΟΡΦΑΝΟΠΟΥΛΟΣ ΧΡΗΣΤΟΣ Π.Π. ΓΕΛ ΕΥΑΓ. ΣΧ. Ν. ΣΜΥΡΝΗΣ [60%]

```
program ishare;

var
        share:text;
        sum,i,n,mo:longint;
        kerdos: array[1 .. 1000000] of longint;
        elegxos:boolean;


begin
        assign(share,'share.in');
        reset(share);
        readln(share,n);
        for i:=1 to n do
        begin
                readln(share,kerdos[i]);
                sum:=sum+kerdos[i];
        end;
        close(share);


        mo:=round(sum/3);
        if mo<sum/3 then
        mo:=mo+1;

        repeat
                elegxos:=true;
                i:=1;
                sum:=0;
                while ((sum+kerdos[i])<=mo) and (n-i>=2) do
                        begin
                        sum:=sum+kerdos[i];
                        i:=i+1;
                        end;

                sum:=0;

        while ((sum+kerdos[i])<=mo) and (n-i>=1) do
                        begin
                        sum:=sum+kerdos[i];
                        i:=i+1;
                        end;

                sum:=0;

                if (i<n) then
                begin
                        for i:=i to n do
                        begin
                        sum:=sum+kerdos[i];
                end;
                end
                else
                        if kerdos[n]>mo then
                        mo:=kerdos[n];

                if sum>mo then
                elegxos:=false;
                mo:=mo+1;

        until elegxos=true;
        mo:=mo-1;
        assign(share,'share.out');
        rewrite(share);
        write(share,mo);
        close(share);
        halt(0);
end.
```

CPP

ΒΟΛΙΚΑΣ ΜΑΡΚΟΣ **Π.Π. ΓΕ.Λ. ΙΩΝΙΔΕΙΟΥ ΣΧΟΛΗΣ ΠΕΙΡΑΙΑ**

```
/*
USER: u1732
LANG: C++
TASK: share
*/

// Volikas Markos

# include <cstdio>
# include <algorithm>
using namespace std;

int N,cum[1000005];

bool can(int m) {
  int first_cut=lower_bound(cum,cum+N,m)-cum;
  if (first_cut==N || cum[first_cut]>m) first_cut--;
  if (first_cut>=N-1) return true;

  int second_cut=lower_bound(cum+first_cut+1,cum+N,cum[first_cut]+m)-
cum;
  if (second_cut==N || cum[second_cut]>cum[first_cut]+m) second_cut--;
  if (second_cut<=first_cut) return false;
  if (second_cut>=N-1) return true;

  if (cum[N-1]-cum[second_cut]>m) return false;

  return true;
  }

inline void fastScan(int &x) {
  register int c=getc_unlocked(stdin);
  x=0;
  for (; (c<48 || c>57); c=getc_unlocked(stdin));
  for (; (c>47 && c<58); c=getc_unlocked(stdin)) { x=(x<<1) + (x<<3) + c-
48; }
  }

int main() {
  freopen("share.in","r",stdin);
freopen("share.out","w",stdout);

int i,a,lo,hi,mid,tmp,ans;

fastScan(N);
fastScan(cum[0]); lo=cum[0];

for (i=1; i<N; ++i) {
    fastScan(a);
    cum[i]=cum[i-1]+a;
    lo=max(lo,a);
    }

hi=cum[N-1];
while (lo<hi) {
    mid=(lo+hi)>>1;
    if (can(mid)) hi=mid;
    else lo=mid+1;
    }

printf("%d\n",hi);

return 0;
}
```

C

ΓΚΑΣΔΡΟΓΚΑΣ ΓΕΩΡΓΙΟΣ ΕΝΙΑΙΟ ΛΥΚΕΙ ΠΛΑΤΥΚΑΜΠΟΥ

```
/* PDP 2015 - 2nd Stage: Senior
 *
 * Contestant Name : Gkasdrogkas Georgios (user: u1778)
 * Problem Name    : Share
 * Language       : C
 *
 * This program finds the minimum profit that can collect the most favored
 * of the three brothers, given the days (N) and the profit of each day (Vi).
 *
 * Max execution time : <= 1 s in worst case (n = 1.000.000)
 * Compexity : O(N) , sooner exit possible using values check (Subroutines 1
& 2)
 *
 * This program use a simple logic. Using two pointers pa and pb pointed to
array[0]
 * and array[N-1] respectively, and using three integers, each for one brother,
we
 * check the next values of the array both for BrotherA and BrotherB. Then
we add the
 * minimum of that two sums to correct brother and proceed the correct
pointer. Then make
 * again the check so in the end to have the result we want.
 *
 * Subroutine 1 : When we pass the numbers, if the maximum number in the
file is greater
 * or equal to all the other sums, then that number is the result we want.
 *
 * Subroutine 2 : If all the profits, in the current state, are equal then we can
safely
 * exit the program.
 */

#include <stdio.h>
#include <stdlib.h>
#include <limits.h>
#include <stdbool.h>

/* Define the min macro and the IO error */
#define min(ProfitA, ProfitB, ProfitC) ProfitA > (ProfitB > ProfitC ? ProfitB :
ProfitC) ? ProfitA : (ProfitB > ProfitC ? ProfitB : ProfitC)
```

```
#define IOERROR 5   // 'Input/Output Error'

FILE *ReadInput(const char *filename_r);
int CountDays(FILE *input);
long PassProfit(FILE *input, long *array, const int N, const char
*filename_w);
long FindProfit(long *array, int N, const char *filename_w);
long SumArray(long *array, int N);
void CheckEqual(const long ProfitA, const long ProfitB, const long ProfitC,
const char *filename_w);
long ProcessMin(const long ProfitA,const long ProfitB,const long ProfitC, long
currentMin, long bestMin, const char *filename_w);
void WriteOutput(const long total, const char *filename_w);

int main() {

    const char *filename_r = "share.in";
    const char *filename_w = "share.out";

    FILE *input = ReadInput(filename_r);

    int N = CountDays(input);

    long *array = malloc(N * sizeof *array);
    if (array==NULL) {
        exit(1);
    }

    PassProfit(input, array, N, filename_w);

    fclose(input);

    long bestMin = FindProfit(array, N, filename_w);

    free(array);
    WriteOutput(bestMin, filename_w);

    return 0;
}

FILE *ReadInput(const char *filename_r) {
    FILE *input = fopen(filename_r, "r");
    if (input == NULL) {
        exit(IOERROR);
    }

    return input;
}

int CountDays(FILE *input) {
    int days = 0;
    fscanf(input, "%d", &days);
    return days;
}

long PassProfit(FILE *input, long *array, const int N, const char *filename_w)
{
    long sum = 0, max = 0;
    for (int i = 0; i < N; i++) {
        fscanf(input, "%ld", &array[i]);
        sum += array[i];
        if (array[i] > max) {
             max = array[i];
        }
    }

    if (max >= sum - max) {             /* If the maximum profit is greater or
equal */
        WriteOutput(max, filename_w);   /* of the other sums (except the max
profit) */
        exit(0);                        /* then that value is what we want!          */
    }

    return *array;
}

long FindProfit(long *array, int N, const char *filename_w) {
    int pA = 0, pB = N - 1;

    long *pa , *pb;
    pa = &array[0];
    pb = &array[N-1];
```

```
long bestMin = LONG_MAX;

long ProfitA = array[0];
long ProfitC = array[N-1];
long ProfitB = SumArray(array, N);

long currentMin = min(ProfitA, ProfitB, ProfitC);

if (currentMin < bestMin) {
    bestMin = currentMin;
}

bool flag = true; // use of 'stdbool.h' header
long check_L = 0, check_R = 0;

while ((pB - pA != 1) && (flag == true)) {
    flag = false;

    *(pa++);                    /* We try to find what are the sums adding */
    *(pb--);                    /* the next index of the array             */
    check_L = ProfitA + *pa;
    check_R = ProfitC + *pb;
    *(pa--);                    // Return the pointers to the previous state
    *(pb++);

    /* Find the minimum of the two sums and add it to correct brother */
    if (check_L <= check_R) {
        *(pa++);
        ProfitA += *pa;
        ProfitB -= *pa;
        pA++;
        flag = true;
    }
    else if (check_L > check_R) {
        *(pb--);
        ProfitC += *pb;
        ProfitB -= *pb;
        pB--;
        flag = true;
    }

    check_R = 0, check_L = 0; // zero the check values
```

```
        bestMin = ProcessMin(ProfitA, ProfitB, ProfitC,currentMin, bestMin,
filename_w);
    }

    return bestMin;
}

long SumArray(long *array, int N) {
    long ProfitB = 0;
    for (int i = 0; i < N; i++) {
        ProfitB += array[i];
    }

    ProfitB = (ProfitB - array[0]) - array[N-1]; // sum of all numbers except the
first and the last element
    return ProfitB;
}

void CheckEqual(const long ProfitA, const long ProfitB, const long ProfitC,
const char *filename_w) {
    if (ProfitA == ProfitC ) {
        if (ProfitB == ProfitA) {
            WriteOutput(ProfitA, filename_w);
            exit(0);
        }
    }
}

long ProcessMin(const long ProfitA,const long ProfitB,const long ProfitC, long
currentMin, long bestMin, const char *filename_w) {
    CheckEqual(ProfitA, ProfitB, ProfitC, filename_w); // check if the three
brothers have the same profit
    currentMin = min(ProfitA, ProfitB, ProfitC); // find the minimum of the
three sums in current state

    if (currentMin < bestMin) {
        bestMin = currentMin; // check if the 'currentMin' is less that the best so
far
    }

    return bestMin;
```

```
}

void WriteOutput(const long total, const char *filename_w) {
    FILE *output = fopen(filename_w, "w");

    if(output == NULL) {
        exit(IOERROR);
    }

    fprintf(output, "%ld\n", total);
    fclose(output);
```