#include <stdio.h>

#include <stdlib.h>

#include <time.h>

int n, m, c;

void swap\_elements(int\* list, int index1, int index2) {

 int temp = list[index1];

 list[index1] = list[index2];

 list[index2] = temp;

}

int partition(int\* list, int start, int end) {

 int i, b;

 i = b = start;

 int p = end;

 while (i < p) {

 if (list[i] <= list[p]) {

 swap\_elements(list, i, b);

 b += 1;

 }

 i += 1;

 }

 swap\_elements(list, b, p);

 p = b;

 return p;

}

int quick\_sort(int\* list, int start, int end) {

 if (end - start < 1)

 return 0;

 int p = partition(list, start, end);

 quick\_sort(list, start, p - 1);

 quick\_sort(list, p + 1, end);

}

int bus\_num(int\* list, int max\_diff) {

 int a = 0; // list[a]~list[b]는 현재 버스에 탈 수 있을지 조사할 때 필요한 사람들(시각), list[a]~list[b-1]은 현재 버스에 탄 사람들

 int b = 1;

 int bus = 1;

 while (b < n) { // list[n-1]까지 존재함

 if (list[b] - list[a] <= max\_diff) { // 대기시간이 최대 대기시간 이하이다.

 if ((b - a + 1) > c) { // 사람 수가 좌석 수에 불만족

 bus++; // 버스 늘림

 a = b; // 다음 버스 처리 시작

 }

 }

 else { // 대기시간이 최대 대기시간 초과이다.

 bus++; // 버스 늘림

 if (bus > m) // 최대 버스 수 초과

 break; // 버스 조사 끝냄

 a = b; // 다음 버스 처리 시작

 }

 b++; // 사람 태움, 다음 경우 조사를 위해 b를 1증가

 }

 return bus;

}

int binary\_search\_compare(int\* list, int first, int last, int fmd) {

 if ((last == 0) || (first == fmd)) // 처음 값 혹은 마지막 값 처리 (최종의 경우), 조건문에서 생략했지만 fisrt=last임

 return first; // 최종의 경우까지 오면 나올 수밖에 없음

 int mid = (first + last) / 2;

 int bus\_mid = bus\_num(list, mid); // 최대 대기시간(max\_diff)이 mid일 때, bus값

 if (mid == 0) { // first:0, last:1이어서 mid가 0이면 'Main Part'의 연산이 불가능 -> 따로 처리

 if (bus\_mid <= m) // max\_diff가 0일 때 조건 만족 -> 0이 최적의 최대 대기시간

 return 0;

 else

 return 1;

 }

 // Main Part

 if (bus\_mid <= m) { // max\_diff가 0이 아닌 mid일 때, bus<=m [현재 조건 만족(적거나 같음)]

 if (bus\_num(list, mid - 1) > m) // max\_diff가 mid일 때, bus>m [바로 전일 때 조건 불만족]

 return mid; // best\_time은 mid

 else // [바로 전일 때 조건 만족]

 binary\_search\_compare(list, first, mid - 1, fmd); // '이전 값들 중 하나'에서 재조사

 }

 else // max\_diff가 0이 아닌 mid일 때, bus>m [현재 조건 불만족]

 binary\_search\_compare(list, mid + 1, last, fmd); // '다음 값들 중 하나'에서 재조사

}

int max\_of\_array(int\* array, int size) {

 int max = array[0];

 int i;

 for (i = 1; i < size; i++) {

 if (array[i] > max)

 max = array[i];

 }

 return max;

}

int main() {

 int i, j;

 double start, finish, duration; // 시간 재기

 // input processing

 printf("n,m,c입력:");

 scanf\_s("%d %d %d", &n, &m, &c);

 // find error

 if (n < c) {

 printf("[n<c] error");

 return 0;

 }

 if (m \* c < n) {

 printf("[mc<n] error");

 return 0;

 }

 // end

 if (n <= 1 || c == 1) {

 printf("0");

 return 0;

 }

 int\* time\_list = (int\*)malloc(sizeof(int) \* n);

 printf("\nn개의 시각 입력:");

 for (i = 0; i < n; i++)

 scanf\_s("%d", &time\_list[i]);

 start = clock(); //시간재기

// sorting

 quick\_sort(time\_list, 0, n - 1);

 printf("\n정렬 결과:");

 for (i = 0; i < n; i++)

 printf("%d ", time\_list[i]);

 printf("\n");

 // make first max\_diff (첫 번째 최대 대기시간(최대 대기차))

 int fmd = 0;

 int tmp = 0;

 int idx = 0; // index of time\_list

 int M = m; // temp\_m

 if ((n % c) == 0) { // n=m\*c, n<m\*c (n = k\*c)

 if ((m \* c - n) >= c) // ----- (m-α) buses can go

 M = n / c;

 printf("M:%d\n", M);

 for (i = 0; i < M; i++) {

 tmp = time\_list[idx + (c - 1)] - time\_list[idx];

 printf("%d번째 시행:[%d,%d] -> %d\n", i, time\_list[idx + (c - 1)], time\_list[idx], tmp);

 idx += c;

 //find max

 if (tmp >= fmd)

 fmd = tmp;

 }

 }

 else { // n<m\*c (n != k\*c)

 if ((m \* c - n) >= c) // ----- (m-α) buses can go

 M = (n / c) + 1;

 printf("M:%d\n", M);

 for (i = 0; i < M; i++) {

 if (i == M - 1) {

 tmp = time\_list[idx + ((n % c) - 1)] - time\_list[idx];

 printf("%d번째 시행:[%d,%d] -> %d\n", i, time\_list[idx + ((n % c) - 1)], time\_list[idx], tmp);

 }

 else {

 tmp = time\_list[idx + (c - 1)] - time\_list[idx];

 printf("%d번째 시행:[%d,%d] -> %d\n", i, time\_list[idx + (c - 1)], time\_list[idx], tmp);

 idx += c;

 }

 //find max

 if (tmp >= fmd)

 fmd = tmp;

 }

 }

 printf("first max\_diff=%d", fmd);

 // find best max\_diff

 int best\_wait = binary\_search\_compare(time\_list, 0, fmd, fmd); // (리스트,first,last,fmd)

 printf("\n\n최적의 최대 대기시간: %d\n\n", best\_wait);

 // 되는 케이스 테스트 하기 [버스 스케줄링]

 int a = 0; // list[a]~list[b]는 현재 버스에 탈 수 있을지 조사할 때 필요한 사람들(시각), list[a]~list[b-1]은 현재 버스에 탄 사람들

 int b = 1;

 int bus = 1; // 버스의 수

 int temp = 0;

 int temp2 = 0;

 best\_wait = best\_wait; // 최대 대기 시간 임의로 정하기 [TEST]

 while (b < n) { // time\_list[n-1]까지 존재함

 if (time\_list[b] - time\_list[a] <= best\_wait) { // 대기시간이 최대 대기시간 이하이다.

 if ((b - a + 1) <= c) // 사람 수가 좌석 수에 만족

 b++; // 버스 태움

 else { // 사람 수가 좌석 수에 만족 X

 printf(" (");

 int i;

 for (i = a; i < b; i++) {

 printf("%d ", time\_list[i]);

 temp++;

 }

 temp2++;

 printf(") %d명 / 대기시간:%d\n", (b - a), time\_list[b - 1] - time\_list[a]);

 bus++; // 버스 늘림

 // (n <= mc이고 가정에 의해, 최대 대기 시간은 만족했는데 버스가 부족해서 못 타는 경우는 존재하지 않는다.)

 a = b; // 버스 늘려서 태움, 다음 버스 처리 시작

 b++;

 }

 }

 else { // 대기시간이 최대 대기시간 초과이다.

 printf(" (");

 int i;

 for (i = a; i < b; i++) {

 printf("%d ", time\_list[i]);

 temp++;

 }

 temp2++;

 printf(") %d명 / 대기시간:%d\n", (b - a), time\_list[b - 1] - time\_list[a]);

 bus++; // 버스 늘림 (m대까지는 커버됨)

 a = b; // 버스 늘려서 태움, 다음 버스 처리 시작

 b++;

 }

 }

 printf(" (");

 int k;

 for (k = a; k < b; k++) {

 printf("%d ", time\_list[k]);

 temp++;

 }

 temp2++;

 printf(") %d명 / 대기시간:%d\n", (b - a), time\_list[b - 1] - time\_list[a]);

 printf("\n n:%d명, 각 버스에 탄 사람들을 합친 수:%d명, m:%d대, 이용 버스:%d대", n, temp, m, temp2);

 if (temp2 > m)

 printf("\n 이용 가능한 버스를 초과하였음!!\n");

 else

 printf("\n 최대 대기 시간은 %d\n", best\_wait);

 finish = clock(); // 시간재기

 duration = (double)(finish - start) / CLOCKS\_PER\_SEC; // 시간재기

 printf("\n\n%.3lf초입니다.\n\n", duration); // 시간재기

}