```c

/**
 * Implementation of a Priority Queue (uses Max-Heap)
 * Author: Mithusayel Murmu
 */

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

#define MAX_PQ_SZ 500

typedef struct _PQueue PQueue;
typedef struct _PQueueElement PQueueElement;

struct _PQueueElement { int key, priority; };
struct _PQueue {
    size_t size;
    PQueueElement data[MAX_PQ_SZ];
};

static inline void swap_data(PQueueElement *a, PQueueElement *b) {
    PQueueElement t = *a; *a = *b; *b = t;
}

/**
 * Recursively max-heapify a node at index idx
 * @queue:      The Priority Queue to use
 * @idx:        Index of node to max-heapify
 */
static void max_heapify(PQueue *queue, size_t idx) {
    int lt = 2 * idx + 1;
    int rt = 2 * (idx + 1);
    int max;

    if (lt < queue->size && queue->data[idx].priority < queue->data[lt].priority)
        max = lt;
    else
        max = idx;
    if (rt < queue->size && queue->data[max].priority < queue->data[rt].priority)
        max = rt;

    if (idx != max) {
        swap_data(queue->data + idx, queue->data + max);
        max_heapify(queue, max);
    }
}

/**
 * Builds a max-heap out of the given array
 * @queue:      The Priority Queue to use
 */
static void build_max_heap(PQueue *queue) {
    int i;
    for (i = queue->size / 2 - 1; i >= 0; i--)
        max_heapify(queue, i);
}

PQueueElement pqueue_extract_max_priority(PQueue *queue) {
    if (queue->size == 0) {
        /* Bogus Priority Queue element */
        PQueueElement elem = { 0, 0 };
        return elem;
    }

    PQueueElement elem = queue->data[0];
    swap_data(queue->data, queue->data + (queue->size - 1));
    queue->size--; max_heapify(queue, 0);

    return elem;
}

/** Parent index */
#define PIDX(x) (((x)-1)/2)

void pqueue_insert_with_priority(PQueue *queue, int _key, int _priority) {
    if (queue->size >= MAX_PQ_SZ) return;

    PQueueElement elem = { .key = _key, .priority = _priority };
    queue->data[queue->size++] = elem;

    int i = queue->size - 1;
    while (i > 0 && _priority > queue->data[PIDX(i)].priority) {
        /* Swap current element with parent */
        swap_data(queue->data + i, queue->data + PIDX(i));
        i = PIDX(i);
    }
}

PQueue * pqueue_create() {
    PQueue *queue = (PQueue*) malloc(sizeof(PQueue));
    queue->size = 0; return queue;
}

/** Driver function */
int main(int argc, char const *argv[]) {
    int N, key, priority;

    printf("Number of integers to use: ");
    scanf("%d", &N);
    printf("Enter %d integers with priorities:\n", N);

    PQueue *queue = pqueue_create();
    while (N--) {
        scanf("%d%d", &key, &priority);
        pqueue_insert_with_priority(queue, key, priority);
    }

    printf("\nPrinting in order of priorities:\n");
    while (queue->size) {
        PQueueElement elem = pqueue_extract_max_priority(queue);
        printf("%d (%d); ", elem.key, elem.priority);
    }
    printf("\n"); free(queue);

    return 0;
}

```