支持优先级的队列题目描述：

实现一个支持优先级的队列，高优先级先出队列；同优先级时先进先出。

如果两个输入数据和优先级都相同，则后一个数据不入队列被丢弃。

队列存储的数据内容是一个整数。

输入描述：

一组待存入队列的数据（包含内容和优先级）

输出描述：

队列的数据内容（优先级信息输出时不再体现）

补充说明：

不用考虑输入数据不合法的情况，测试数据不超过 100 个

示例 1

输入：

```
(10,1),(20,1),(30,2),(40,3)
```

输出：

```
40,30,10,20
```

说明：

输入样例中，向队列写入了 4 个数据，每个数据由数据内容和优先级组成。

输入和输出内容都不含空格。

数据 40 的优先级最高，所以最先输出，其次是 30；10 和 20 优先级相同，所以按输入顺序输出。

示例 2

输入：

```
(10,1),(10,1),(30,2),(40,3)
```

输出：

```
40,30,10
```

说明：

输入样例中，向队列写入了 4 个数据，每个数据由数据内容和优先级组成。

输入和输出内容都不含空格。

数据 40 的优先级最高，所以最先输出，其次是 30；两个 10 和 10 构成重复数据，被丢弃一个。

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <stdbool.h>

typedef struct {
	int a;
	int b;
} item;

int cmp(const void *x, const void *y) {
	return ((item *) y)->b - ((item *) x)->b;
}

int main() {
	char str[200] = {0};
	gets(str);
	int len = strlen(str);
	char tgt[200] = {0};
	int tgt_size = 0;
	for (int i = 0; i < len; ++i) {
		if (str[i] == '(' || str[i] == ')') {
			continue;
		}
		tgt[tgt_size++] = str[i];
	}
	char *queues[200] = {0};
	int queues_size = 0;
	char *tok = strtok(tgt, ",");
	while (tok != NULL) {
		queues[queues_size++] = tok;
		tok = strtok(NULL, ",");
	}
	item items[200] = {0};
	int item_size = 0;
	for (int i = 0; i < queues_size; i += 2) {
		int a = atoi(queues[i]);
```

```
            int b = atoi(queues[i + 1]);
            bool flag = false;
            for (int j = 0; j < item_size; ++j) {
                if (items[j].a == a && items[j].b == b) {
                    flag = true;
                    break;
                }
            }
            if (!flag) {
                items[item_size].a = a;
                items[item_size].b = b;
                item_size++;
            }
        }

        qsort(items, item_size, sizeof(item), cmp);

        int res[1000] = {0};
        int m = 0;
        for (int i = 0; i < queues_size; ++i) {
            if(items[i].a != 0){
                res[m++] = items[i].a;
            }
        }

        for(int i = 0; i < m - 1; ++i){
            printf("%d,", res[i]);
        }
        printf("%d", res[m - 1]);
        return 0;
}
```