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

using namespace std;

typedef struct ListNode{
    int cost;
    struct ListNode * next;
}ListNode;

int Cmp(const void * a,const void * b){
    int *arr1 = *(int **)a;
    int *arr2 = *(int **)b;
    if(arr1[0] == arr2[0]){
        return arr1[1] - arr2[1];
    }
    return arr1[0] - arr2[0];
}

ListNode * addList(ListNode * head,int cost){
    if(!head || head->cost > cost){
        ListNode * node = (ListNode *)malloc(sizeof(ListNode));
        node->cost = cost;
        node->next = head;
        return node;
    }

    head->next = addList(head->next,cost);
    return head;
}

ListNode * removeList(ListNode * head){
    if(!head) return NULL;
    ListNode * newHead = head->next;
    free(head);
    return newHead;
}

void debugList(ListNode * list){
    for(ListNode * node = list ; node ; node = node->next){
        printf("%d ",node->cost);
    }
    printf("\n\r");
}

int AllTime(int ** requests,int requestSize,int * requestColSize){
    int wait = 0;
    int curr = 0;
    int total = 0;
    long long waitTask = 0;
    int mod = 1e9 + 7;
```

```c
    ListNode * list = NULL;
    qsort(requests,requestSize,sizeof(int *),Cmp);
    curr = requests[0][0];
    for(int i = 0; i < requestSize; ++i){
        //printf("(%d,%d) \n\r",requests[i][0],requests[i][1]);
        //printf("total = %d \n\r",total);
        //printf("wait task = %d\n\r",waitTask);
        //printf("curr = %d\n\r",curr);
        //debugList(list);
        if(!list || requests[i][0] <= curr){
            list = addList(list,requests[i][1]);
            waitTask++;
            continue;
        }

        if(requests[i][0] > curr){
            while(!list && curr + list->cost <= requests[i][0]){
                curr = curr + list->cost;
                total = (total + list->cost*(waitTask-1))%mod;
                list = removeList(list);
                waitTask--;
            }

            if(list){
                wait = requests[i][0] - curr;
                total = (total + wait*(waitTask-1))%mod;
                int rest = list->cost - wait;
                list = removeList(list);
                waitTask--;
                list = addList(list,rest);
                waitTask++;
            }
            curr = requests[i][0];
            list = addList(list,requests[i][1]);
            waitTask++;
        }
    }

    while(list){
        total = (total + list->cost*(waitTask-1))%mod;
        list = removeList(list);
        waitTask--;
    }

    return total;
}

int main(){
    int ** task1 = (int **)malloc(sizeof(int *)*3);
    int ** task2 = (int **)malloc(sizeof(int *)*4);
    int col = 2;

    for(int i = 0; i < 3; ++i){
        task1[i] = (int *)malloc(sizeof(int)*2);
    }
    for(int i = 0; i < 4; ++i){
        task2[i] = (int *)malloc(sizeof(int)*2);
    }
    task1[0][0] = 0;
    task1[0][1] = 5;
    task1[1][0] = 1;
    task1[1][1] = 2;
    task1[2][0] = 2;
    task1[2][1] = 1;

    task2[0][0] = 0;
    task2[0][1] = 5;
    task2[1][0] = 1;
    task2[1][1] = 2;
    task2[2][0] = 0;
    task2[2][1] = 2;
    task2[3][0] = 1;
    task2[3][1] = 2;
    printf(" total wait = %d\n\r",AllTime(task1,3,NULL));
    printf(" total wait = %d\n\r",AllTime(task2,4,NULL));
}
```