int cmp(const void \*a,const void \*b){

    return \*(int\*)a -\*(int\*)b;

}

/\*\*

 \* Return an array of arrays of size \*returnSize.

 \* The sizes of the arrays are returned as \*returnColumnSizes array.

 \* Note: Both returned array and \*columnSizes array must be malloced, assume caller calls free().

 \*/

int\*\* threeSum(int\* nums, int numsSize, int\* returnSize, int\*\* returnColumnSizes){

    int unit=512;

    int\*\* ans=(int \*\*)malloc(sizeof(int\*)\*unit);

    \* returnColumnSizes=NULL;

    \* returnColumnSizes=(int \*)realloc(\*returnColumnSizes,sizeof(int)\*unit);

    \* returnSize=0;

    int i=0;

    int k=i+1;

    int j=numsSize-1;

    qsort(nums,numsSize,sizeof(int),cmp);

    while(1){

        int sum=nums[i]+nums[k]+nums[j];

        if( sum == 0){//確認是否sum==0;

            (\* returnSize)++;

            if( (\* returnSize) % unit == 0){

                ans=realloc(ans,sizeof(int\*)\*(\* returnSize+unit));

                \* returnColumnSizes=realloc(\*returnColumnSizes,sizeof(int)\*(\* returnSize+unit));

            }

            ans[\* returnSize -1]=(int \*)malloc(sizeof(int)\*3);

            ans[\* returnSize -1][0]=nums[i];

            ans[\* returnSize -1][1]=nums[k];

            ans[\* returnSize -1][2]=nums[j];

            (\* returnColumnSizes)[\* returnSize -1]=3;

            do{

            j--;

            k++;

            }while(  j > 1 && k < numsSize-2 && nums[j]==nums[j+1] && nums[k]==nums[k-1] );

        }else if( sum > 0){

                j--;

        }else if( sum < 0){

                k++;

        }

        if( j <= k){

            do{

                i++;

                k=i+1;

                j=numsSize-1;

            }while(  i < numsSize-2 && nums[i]==nums[i-1] );

        }

        if( k==j ){break;}

    }

    return ans;

}