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

  return \*(int\*)a - \*(int\*)b;

}

typedef struct {

  int count;

  int topnum;

}Pair;

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

  const Pair \*pa=(const Pair\*)a;

  const Pair \*pb=(const Pair\*)b;

  return pa->count-pb->count;

}

/\*\*

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

 \*/

int\* topKFrequent(int\* nums, int numsSize, int k, int\* returnSize){

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

//for(int i=0;i<numsSize;i++){

//  printf("%d ",nums[i]);

//}

//printf("\n");

Pair \*pair=malloc(sizeof(Pair)\*k);

int \*result=NULL;

int c=0;

int current=nums[0];

int index=0;

if( numsSize == 1){

    pair[0].topnum=nums[0];

}else{

  for(int i=0 ; i<numsSize ; i++){

    //printf("current:%d ,num[i]=%d \n",current,nums[i]);

    if( current == nums[i] ){

      c++;

    }

    if( current != nums[i] || i==numsSize-1){

      if(index < k){

            pair[index].count=c;

            pair[index].topnum=current;

            c=1;

            current = nums[i];

            index++;

      }else{

          qsort(pair,k,sizeof(Pair),cmpPair);

          //printf("c=");

          for(int i=0;i<k;i++){

            //printf("%d(%d) ",pair[i].count,pair[i].topnum);

          }

          //printf("\n");

          for(int j=0;j<k;j++){

            //printf("1 current:%d count:%d: topnum=%d c=%d\n",current,pair[j].count,pair[j].topnum,c);

            if(pair[j].count<c){

              //printf("2 current:%d count:%d: topnum=%d c=%d\n",current,pair[j].count,pair[j].topnum,c);

              pair[j].count=c;

              pair[j].topnum=current;

              //printf("change top:%d\n",pair[j].topnum);

              c=1;

              current = nums[i];

              break;

            }

        }

        c=1;

        current = nums[i];

      }

    }

    if( i==numsSize-1){

      for(int j=0;j<k;j++){

        //printf("1county:%d:%d\n",pair[j].count,c);

        if(pair[j].count<c){

          //printf("2county:%d:%d\n",pair[j].count,c);

          pair[j].count=c;

          pair[j].topnum=current;

          //printf("top:%d\n",pair[j].topnum);

          c=1;

          current = nums[i];

          break;

        }

      }

    }

  }

}

//printf("\n");

result=malloc(sizeof(int)\*(k));

for(int i=0;i<k;i++){

  result[i]=pair[i].topnum;

  //printf("top:%d ",result[i]);

}

\* returnSize=k;

//printf("\*returnSize=%d",\*returnSize);

return result;

}