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

    return \*(const char \*)a-\*(const char \*)b;

  }

typedef struct {

  char \*original\_str;

  char \*sorted\_str;

}Pair;

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

  const Pair \*pa = (const Pair\*)a;

  const Pair \*pb = (const Pair\*)b;

  return strcmp(pa->sorted\_str,pb->sorted\_str);

}

/\*\*

 \* 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().

 \*/

char \*\*\* groupAnagrams(char \*\* strs, int strsSize, int\* returnSize, int\*\* returnColumnSizes){

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

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

  char \*sorted  =malloc(sizeof(char)\*(strlen(strs[i])+1));

  strcpy(sorted,strs[i]);//將排序前的紀錄

  qsort(sorted,strlen(strs[i]),sizeof(char),cmp);//排序

  pair[i].original\_str=strs[i];

  pair[i].sorted\_str=sorted;

  //printf("%s:%s\n",pair[i].original\_str,pair[i].sorted\_str);

}

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

/\*\*\*

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

  printf("%d ",i);

  printf("%s :%s\n",pair[i].original\_str,pair[i].sorted\_str);

}

\*\*\*/

//bat :abt

//eat :aet

//tea :aet

//ate :aet

//tan :ant

//nat :ant

//[["bat"](群組一)),["nat","tan"](群組二),["ate","eat","tea"](群組三)]

//--------------初始化---------------

char \*\*\*result=NULL;

\* returnSize = 0;

\* returnColumnSizes=NULL;

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

  printf("%d \n",i);

  //printf("%s :%s\n",pair[i].original\_str,pair[i].sorted\_str);

  if(i==0 || strcmp(pair[i].sorted\_str,pair[i-1].sorted\_str)!=0){//--------------增加一個群組---------

    printf("%d \n",i);

    int LastGruopIndex=\* returnSize;

    result=realloc(result,sizeof(char\*\*)\*(\* returnSize+1));//新增一個空間給result

    result[LastGruopIndex]=malloc(sizeof(char\*)\*1);//在群組中增加一個字串空間

    result[LastGruopIndex][0]=pair[i].original\_str;//在群組中第一個位置塞入字串

    (\* returnSize)++;//增加一個群組

    \* returnColumnSizes=realloc(\* returnColumnSizes,sizeof(int)\*(\* returnSize));//增加索引群組int的空間

    (\*returnColumnSizes)[LastGruopIndex]=1;//增加索引(一個群組內有多少字串)

  }else{//在群組下增加字串

    int LastGroupIndex= \*returnSize-1;//要退回上一個group(如果沒有-1就會新增下一個group)，取出目前有多少群組

    int LastGroupSize= (\*returnColumnSizes)[LastGroupIndex];//取出將最後一個群組的空間大小

    result[LastGroupIndex]=realloc(result[LastGroupIndex],sizeof(char\*)\*(LastGroupSize+1));//新增一個字串空間給result[LastGroupIndex]

    result[LastGroupIndex][LastGroupSize]=pair[i].original\_str;//在群組中塞入字串

    (\*returnColumnSizes)[LastGroupIndex]=LastGroupSize+1;//增加索引(一個群組內有多少字串)

  }

}

/\*\*字串矩陣測試\*\*

char \*\*\* result = malloc(sizeof(char\*\*)\*3);

result[0] = malloc(sizeof(char\*)\*1);

result[1] = malloc(sizeof(char\*)\*2);

result[2] = malloc(sizeof(char\*)\*3);

result[0][0] = malloc(sizeof(char)\*(strlen("bat")+1));

result[0][0]="bat";

result[1][0] = malloc(sizeof(char)\*(strlen("nat")+1));

strcpy(result[1][0],"nat");

result[1][1] = malloc(sizeof(char)\*(strlen("tan")+1));

strcpy(result[1][1],"tan");

result[2][0] = malloc(sizeof(char)\*(strlen("ate")+1));

strcpy(result[2][0],"ate");

result[2][1] = malloc(sizeof(char)\*(strlen("eat")+1));

strcpy(result[2][1],"eat");

result[2][2] = malloc(sizeof(char)\*(strlen("tea")+1));

strcpy(result[2][2],"tea");

\*returnSize=3;

\*returnColumnSizes = malloc(sizeof(int)\*3);

  (\*returnColumnSizes)[0]=1;

  (\*returnColumnSizes)[1]=2;

  (\*returnColumnSizes)[2]=3;

return result;

\*\*\*/

return result;

}