```
#if 0
void bt(int *buff, int ***p, int *psz, int *pn, int n, int k, int d, int start) {
    int i;
    if (d == k) {
        // all done
        if (*psz == *pn) {
            *psz *= 2;
            *p = realloc(*p, *psz * sizeof(int *));
        }
        (*p)[*pn] = malloc(k * sizeof(int));

        memcpy((*p)[(*pn) ++], buff, k * sizeof(int));
        return;
    }
    for (i = start; i <= n; i ++) {
        buff[d] = i;
        bt(buff, p, psz, pn, n, k, d + 1, i + 1);
    }
}
int** combine(int n, int k, int* returnSize, int** returnColumnSizes) {

    int **p, *buff;
    int psz, pn;

    pn = 0;
    psz = 10;
    p = malloc(psz * sizeof(int *));
    buff = malloc(k * sizeof(int));

    bt(buff, &p, &psz, &pn, n, k, 0, 1);

    free(buff);

    *returnColumnSizes = malloc(pn * sizeof(int));

    *returnSize = pn;

    while (pn --) {
        (*returnColumnSizes)[pn] = k;
    }

    return p;
}

#else
#define ALLOC_LENGTH        (10)
int alloc_length = ALLOC_LENGTH;
int** result;

void _combine(int start, int n, int k, int pos, int* returnSize, int** returnColumnSizes)
{
    static int tmp[20];
    int index;
    int tmp_alloc_length;

    if(pos == k)
    {
        memcpy(result[*returnSize], tmp, sizeof(int)*k);
        (*returnColumnSizes)[*returnSize] = k;
        (*returnSize)++;


        if( alloc_length == *returnSize )
        {
            tmp_alloc_length = alloc_length;
            alloc_length *= 2;
            *returnColumnSizes = (int*)realloc(*returnColumnSizes, sizeof(int)* alloc_length);
             result = (int**)realloc(result, sizeof(int*)* alloc_length);
             result[(*returnSize)] = (int*)malloc( sizeof(int) * tmp_alloc_length*k);
        }else
        {
            result[*returnSize] = result[(*returnSize) - 1] + k;
        }
        return ;
    }

    for(index = start; index <= n; index++)
    {
        tmp[pos] = index;

        _combine(index+1, n, k, pos+1, returnSize, returnColumnSizes);

    }
}


int** combine(int n, int k, int* returnSize, int** returnColumnSizes) {
    int index;
    int count;
    int tmp_index;
    int remain;

    *returnSize = 0;
    *returnColumnSizes = (int*)malloc( sizeof(int)* alloc_length);
    result = (int**)malloc( sizeof(int*) * alloc_length);
    result[0] = (int*)malloc( sizeof(int) * alloc_length*k);
```

```
    _combine(1, n, k, 0, returnSize, returnColumnSizes);

    return result;
}
#endif
```