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

void swap(char *a, char *b)
{
	char temp = *a;
	*a = *b;
	*b = temp;
}

int partition(char *arr, int start, int end, char pivot)
{
	int index = start;
	for(int index2 = start; index2 < end; index2++)
	{
		if(arr[index2] < pivot)
			swap(&arr[index2], &arr[index++]);
		else if(arr[index2] == pivot)
			swap(&arr[index2--], &arr[end]);
	}
	swap(&arr[index], &arr[end]);
	return index;
}


//function work similar to quick sort
void matchNutsBolts(char *nuts, char *bolts, int start, int end)
{
	if(start < end)
	{
		int pivot = partition(nuts, start, end, bolts[end]);

		//bolts partion
		partition(bolts, start, end, nuts[pivot]);

		matchNutsBolts(nuts, bolts, start, pivot-1);
		matchNutsBolts(nuts, bolts, pivot+1, end);
	}
}

void printArray(char *arr, int size)
{
	printf("\n");
	for(int index = 0; index < size; index++)
		printf("%c\t", arr[index]);
}

int main()
{
	int index, size;
	char *nuts, *bolts;
	printf("Enter size of nuts or bolts\n");
```

```
        scanf("%d", &size);

        //allocate memory
        nuts = (char *)malloc(sizeof(char) * size);
        bolts = (char *)malloc(sizeof(char) * size);

        printf("Enter characters of nuts\n");
        for(index = 0; index < size; index++)
                scanf("%s", &nuts[index]);

        printf("Enter characters of bolts\n");
        for(index = 0; index < size; index++)
                scanf("%s", &bolts[index]);

        matchNutsBolts(nuts, bolts, 0, size-1);

        printf("Matched nuts and bolts are \n");
        printArray(nuts, size);
        printArray(bolts, size);
        return 0;
}
```

Time complexity: O(nlogn)
Space complexity: O(1)