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

struct node
{
	int data;
	struct node *next;
};

struct node *newNode(int data)
{
	struct node *temp = (struct node *)malloc(sizeof(struct node));
	temp->data = data;
	temp->next = NULL;
	return temp;
}

struct node *MergeSortedList(struct node *list1, struct node *list2)
{
	struct node *result = NULL;
	if(!list1)
		return list2;
	if(!list2)
		return list1;

	if(list1->data <= list2->data)
	{
		result = list1;
		result->next = MergeSortedList(list1->next, list2);
	}
	else
	{
		result = list2;
		result->next = MergeSortedList(list1, list2->next);
	}
	return result;
}

struct node *mergeKsortedLists(struct node *arr[], int last)
{
	while(last)
	{
		int start = 0, end = last;
		while(start < end)
		{
			arr[start] = MergeSortedList(arr[start++], arr[end--]);
			if( start >= end)
				last = end;
		}

	}
	return arr[0];
```

```
}

void printList(struct node *head)
{
	for(; head; head = head->next)
		printf("%d\t", head->data);
}

int main()
{
	int k = 3, size = 4;

	struct node *arr[k];

	arr[0] = newNode(1);
	arr[0]->next = newNode(3);
	arr[0]->next->next = newNode(5);
	arr[0]->next->next->next = newNode(7);

	arr[1] = newNode(2);
	arr[1]->next = newNode(4);
	arr[1]->next->next = newNode(6);
	arr[1]->next->next->next = newNode(8);

	arr[2] = newNode(0);
	arr[2]->next = newNode(9);
	arr[2]->next->next = newNode(10);
	arr[2]->next->next = newNode(11);
	struct node *head = mergeKsortedLists(arr, k - 1);
	printList(arr[0]);
	return 0;
}
```

Time complexity: O(nk logk)
Space complexity: O(n)