Examples and Exercises

Question 1
Given 2 ascending order lists, a and b, how to merge them into the ascending order list c?

- \( a = \{0, 2, 3, 8\} \)
- \( b = \{1, 4, 5, 7, 9, 10\} \)
- \( c = \{0, 1, 2, 3, 4, 5, 7, 8, 9, 10\} \)

Let’s do it by a user-defined function.

Examples and Exercises

Question 2
So far, we have introduced 3 sorting algorithms: selection, insertion, and bubble. The complexity is \( O(n^2) \) for each of them.

Any \( O(n\log n) \) sorting algorithm?
Examples and Exercises

Recursion function
A function that calls itself is called recursion function.

Recursion definition
1. Every recursive definition must have one (or more) base cases.
2. The general case must eventually be reduced to a base case.
3. The base case stops the recursion.

Examples and Exercises

Another Example

0! = 1
n! = n*(n-1)! n>0

Write a recursive function to calculate n!

Examples and Exercises

void mSort(int list[], int length)
{
    int mid = length / 2, temp;
    int list1[mid]; int list2[length - mid];
    if(length >= 3)
    {
        for(int i = 0; i <= mid - 1; i++)
            list1[i] = list[i]; //there are mid elements in list1
        for(int i = 0; i <= length - mid - 1; i++)
            list2[i] = list[mid + i]; //there are length-mid elements in list2
        mSort(list1, mid);
        mSort(list2, length-mid);
        merge(list1, mid, list2, length-mid, list);
    }
    else if(length == 2)
    {
        if(list[0] > list[1])
        {
            temp = list[1];
            list[1] = list[0];
            list[0] = temp;
        }
    }
}

void merge(int list[], int first, int mid, int last)
{
    int i = first, j = mid + 1, k = first;
    while(i <= mid && j <= last)
    {
        if(list[i] <= list[j])
            list[k++] = list[i++];
        else
            list[k++] = list[j++];
    }
    while(i <= mid)
        list[k++] = list[i++];
    while(j <= last)
        list[k++] = list[j++];

Examples and Exercises

void mergeSort(int list[], int length)
{
    int mid = length / 2, temp;
    int list1[mid]; int list2[length - mid];
    if(length >= 3)
    {
        for(int i = 0; i <= mid - 1; i++)
            list1[i] = list[i]; //there are mid elements in list1
        for(int i = 0; i <= length - mid - 1; i++)
            list2[i] = list[mid + i]; //there are length-mid elements in list2
        mergeSort(list1, mid);
        mergeSort(list2, length-mid);
        merge(list1, mid, list2, length-mid, list);
    }
    else if(length == 2)
    {
        if(list[0] > list[1])
        {
            temp = list[1];
            list[1] = list[0];
            list[0] = temp;
        }
    }
}

Examples and Exercises

void mergeSort2(int list[], int length)
{
    int mid = length / 2, temp;
    int list1[mid]; int list2[length - mid];
    if(length >= 3)
    {
        for(int i = 0; i <= mid - 1; i++)
            list1[i] = list[i]; //there are mid elements in list1
        for(int i = 0; i <= length - mid - 1; i++)
            list2[i] = list[mid + i]; //there are length-mid elements in list2
        mergeSort2(list1, mid);
        mergeSort2(list2, length-mid);
        merge(list1, mid, list2, length-mid, list);
    }
    else if(length == 2)
    {
        if(list[0] > list[1])
        {
            temp = list[1];
            list[1] = list[0];
            list[0] = temp;
        }
    }
}
Examples and Exercises

And thus write the code for merge-sort algorithm

```cpp
void mSort(int list[], int first, int last)
```

mergeSort2.cpp