SSZG519 Data Structures & Algorithms Design
EC-1 Assignment - 1
Weightage 7% Max marks : 7M

IMPORTANT:

- You must only upload the source files for the 1 problem.
- All code related to problem 1 must be inside a folder named q1
- You must provide a text file (.txt) named **README** clearly indicating how to compile and execute your code inside the directory.
- The names of the ADTs and functions suggested must be maintained as it is.

Note: C-like function prototypes are given here for your convenience. End Note.

Problem 1 [To be written inside a folder named q1]

[This problem may require additional functions. If you feel that more functions are required, you may write them leaving a note in the README file.]

Write a program that finds the kth smallest Number from a list of any base-n numbers (for any n up to a fixed maximum value, maxN). The definition of the ADT called `Number` is the same as that defined in Lab-1. For ease of implementation, assume that each digit is represented by a character. A separate lookup table needs to be created for storing the symbols (character) corresponding to the various digit values.

The program must take two command-line parameters:
the first parameter is the name of a file containing the sequence of characters to be used for the digit values from 0 to maxN-1, and
the second parameter is the name of a file containing N Numbers.

The sequence of characters corresponding to digit values 0 to maxN-1 is given in the first line of the first input file, separated by spaces. Note that maxN to be used in the particular instance of execution is calculated by the number of entries in the first line. Only alphanumeric characters can be used for representing digits in the number system. (Note that the total number of alphanumeric characters is limited to 62.)
Consider this sample file:

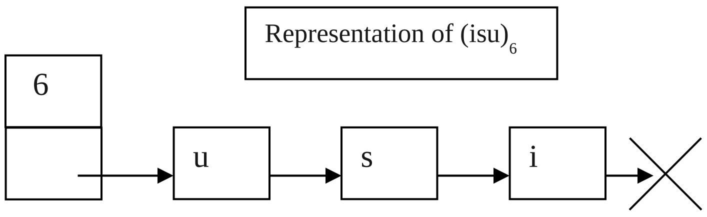
File1:
```
b i g s o u n d
```

By reading this you should infer that **maxN** is 8 and the value of symbol **`o`** is 4 and that of symbol **`n`** is 6 and so on.

The second input file contains a list of N Numbers, with one Number in each line.
Consider the following sample (second) file: containing the numbers 59 in base-6, 36 in base-8, 10 in base-2, 7 in base-3, 576 in base-4, 9 in base-2. Representation of the first number in File2 is shown below.

File2:

```
6 isu
8 go
2 ibib
3 gi
4 gibbs
2 ibbi
```



Type definitions:

The definition for lookup table named **Base**, containing a list of symbols indexed by digit values. Note that since the base is immutable (i.e. won't change), the list can be allocated at time of definition.

The type definition for a **Number**, containing the fields: the base of the number, a linked list of digits each of which is a character, and other field that may be useful.

The type definition for an enumerated type called **Order**, that can have either **GREATER, LESSER** or **EQUAL**.

Design of **Base** ADT with the following operations

(a) **int initializeBase(FILE *basefile)** : reads the file containing the characters corresponding to each digit, populates the lookup table and returns the maximum base supported in the current execution. Note that to use the number system; this function has to be called first, before any other function.

(b) **int lookup(char c)** : looks-up the character in the **Base** lookup table and returns the value associated with the character.

Design of **Number** ADT with the following operations:

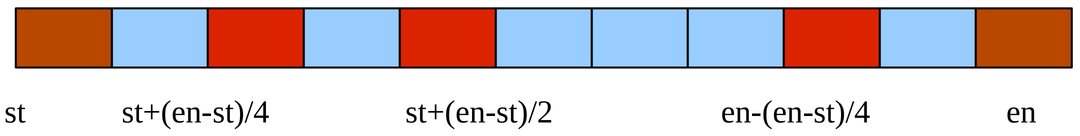
(a) **Number createNumber(char *number_format)**: takes the number format as a string containing the base of the number, followed by a space, followed by the number using the custom characters(Most Significant Digit first). This function then creates a **Number** using the values corresponding to the digits and returns it.

(b) **void printNumber(Number n):** prints the **Number** in the base stored in the system.

(c) **Number convert(Number n, int to_base)**: converts the number **n** to a number of base **to_base**, and returns it.

(d) **Order compare(Number n1, Number n2):** This function is used to compare two Numbers of any valid base. This function returns **GREATER** if n1 > n2**;** **LESSER** if n1 < n2; or **EQUAL** if n1 = n2. This function must be written using recursion.

(e) **int partition(Number list[], int start, int end):** This function partitions the **list[start...end]** into 2 sublists, such that all Numbers in left sublist is less than a chosen pivot and all the Numbers in the right sublist are greater than the chosen pivot. Here, the pivot element is chosen as median of the five elements **`list[start]`, `list[start+(end-start)/4]`, `list[end-(end-start)/4]`, `list[start+(end-start)/2]`,** and **`list[end]`**. This function uses **`compare`** described above for comparison.



(f) **`Number quickSelect(Number list[], int size, int k):`** This functions returns the kth smallest Number in the list, using an iterative version of quickselect. This function uses **`partition`** described above for partitioning about a pivot element,

Driver file:
Create a simple driver file that reads the two input files, the value of k, and the name of the output file as command-line arguments.
The first input file containing the characters used in the number systems. The second input file must contain the list of Numbers to be used for finding the kth smallest element. The driver should generate a dynamic array of numbers. The output file must contain the kth smallest number in the format:

```
(number)base
```

For the above sample input files(File1 and File2), and value of k=3, the program **`kthsmallest`** must print the following in the output file.

```
(ibib)2
```

The driver file must read all the Numbers from the first input file and store it as a dynamic array of Numbers. The size of the array can be computed as the number of lines in the input file. If k > size of array(N), then take k as N/2. Then, the driver calls **`quickSelect`** function to determine the kth smallest element. Then the code should print the output in output file specified.

Tasks to perform:
1. Implement the functions **`quickSelect`, `partition`** and **`compare`** described above [2+1+2= 5 M]
2. Implement the driver file, and give instructions on how to compile and execute your code and store the results in output file. [2 M]