Programming Guidelines

1. Use comments to explain your codes
   - 1.1 Header comments at the beginning of a file
   - 1.2 Global variables and function declarations need to have comments
   - 1.3 Key operations must be clearly documented.
   - 1.4 Spelling must be correct.
   - 1.5 Comments should also be properly indented and with space char inserted.
2. Indentation to group statements at the same block level
   - 2.1 Use `<tab>` for indentation.
3. Blank lines to separate
   - 3.1 directives and functions
   - 3.2 declarations and statements
   - 3.3 All declarations must precede statements in a function.
4. Space character to separate tokens
   - 4.1 The same way as in English sentences
5. Variable name should be descriptive.
   - 5.1 `i, j, k` for integral local variables
   - 5.2 `x, y, z` for `floating point` local variables
   - 5.3 `p, q, r` for local `pointers`
   - 5.4 `All-capital` tokens for symbolic constants
6. Each line of source code should not have more than 80 characters.

Example

```
// EE3980 HW01 Quadratic Sorts
// ID, 姓名
// 2020/03/10

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/time.h>

int N;                                  // input size
char **data;                            // input data
char **A;                               // array to be sorted
int R = 500;                            // number of repetitions

void readInput(void);                   // read all inputs
void printArray(char **A);              // print the content of array A
void copyArray(char **data, char **A); // copy data to array A
double GetTime(void);                   // get local time in seconds
void SelectionSort(char **list,int n); // in-place selection sort
void InsertionSort(char **list,int n); // in-place insertion sort
void BubbleSort(char **list,int n);    // in-place bubble sort
void ShakerSort(char **list,int n);    // in-place shaker sort
```

Example, II

```
int main(void)
{
    int i;                              // loop index
    double t;                           // for CPU time tracking

    readInput();                        // read input data
    t = GetTime();                      // initialize time counter
    for (i = 0; i < R; i++) {
        copyArray(data, A);             // initialize array for sorting
        SelectionSort(A, N);            // execute selection sort
        if (i == 0) printArray(A);      // print sorted results
    }
    t = (GetTime() - t) / R;            // calculate CPU time per iteration
    printf(" .... ", t);                // print out CPU time

    return 0;
}
```