Practice #2

Algorithm Specification ~ Recursion

Yunmin Go

School of CSEE

HANDONG GLOBAL UNIVERSITY

Practice #2 TO-DO List

To-Do	Submission	Notes
String Conversion	Screenshot and source code (strconv.c)	p.34, Chapter 1
Selection Sort	Screenshot and source code (selsort.c)	p.38 ~ p.41, Chapter 1
Binary Search (Recursive)	Screenshot and source code (binsearch.c)	p.43~p.47, p.53~55, Chapter 1

- Upload your screenshot and source codes on LMS by 11pm on 3/9 (Tue).
  - All your screenshots should be merged in one pdf file, screenshot.pdf.
  - Your pdf and all source codes should be compressed into zip file.
  - Your zip file contains four files, screenshot.pdf, strconv.c, selsort.c, and binsearch.c.
- File name: practice02_Your Student ID_Name.zip (only zip, not pdf, docx, c, etc)
  - ex) practice02_20400022_고윤민.zip

String Conversion (1/2)

- Implement a function convert_case() to convert letter cases
  - Skeleton code: *strconv.c*
  - Refer to p.32, chapter 1.
  - Do not use any character array in convert_case() → use malloc()
  - Hint
    - strlen(): get length of string
    - isupper(): checks for an upper case letter
    - islower(): checks for a lower case letter
    - tolower(): convert letter to lower case
    - toupper(): convert letter to upper case

```
#define MAX_LEN 1024
char* convert_case(char*);
int main()
{
    char aLine[MAX_LEN];
    char *p;

    fgets(aLine, MAX_LEN, stdin);

    p = convert_case(aLine);
    printf("%s□n", p);

    free(p);
    return 0;
}
```

String Conversion (2/2)

- Expected results

```
PS C:\ds\practice02> .\strconv.exe
Hello World
hELLO wORLD

PS C:\ds\practice02> .\strconv.exe
hanDONG Global uNivErsity
HANdong gLOBAL UnIVeRSITY
```

Selection Sort (1/2)

- Implement a program that sort natural numbers by descending order using selection sort algorithm
  - Skeleton code: *selsort.c*
  - Refer to p.38~42, chapter 1.
  - Procedures of main() in *selsort.c*
    - Step 1. Read numbers from file
    - Step 2. Sort data by descending order (ex. 10, 9, 8, 7, 6, 5….)
    - Step 3. Save sorted numbers into file
  - Complete read_data(), sort_data(), exchange()
    - read_data(): determine the number of natural numbers (i.e., *num*) stored in file and make array (i.e., *data*) by reading numbers from file
    - sort_data(): sort *num* natural numbers by descending order using selection sort
    - exhange(): exchange value 1 and value 2 for sorting

Selection Sort (2/2)

- Expected results

```
PS C:\ds\practice02\sol> .\selsort.exe
After read: num=20
5985 7973 475 678 2769 9454 2044 7094 3771 6721 546 885 838 4759 7346 744 9583 175 9765 8166

After sort: num=20
9765 9583 9454 8166 7973 7346 7094 6721 5985 4759 3771 2769 2044 885 838 744 678 546 475 175
```

Binary Search (1/2)

- Implement a program that finds an index of search number from the sorted list by using binary search algorithm
  - Skeleton code: *binsearch.c*
  - Refer to p.43~47 and p.53~55, chapter 1.
  - Procedures of main() in *binsearch.c*
    - Step 1. Read numbers from file (numbers are sorted by descending order already)
    - Step 2. Enter a search number and find an index of that from the sorted array
    - Step 3. Repeat step 2 until the entered search number is less than 0
  - Complete read_data(), binsearch(), compare()
    - read_data(): same with read_data() in *selsort.c*
    - binsearch (): find an index of search number using binary search algorithm (data is sorted by descending order)
    - compare(): compare two values for binary search

Binary Search (2/2)

- Expected results

```
PS C:\ds\practice02\sol> .\binsearch.exe
After read: num=20
9765 9583 9454 8166 7973 7346 7094 6721 5985 4759 3771 2769 2044 885 838 744 678 546 475 175

Search number: 9454
The search number is at 2
Search number: 9450
There is no search number
Search number: 6721
The search number is at 7
Search number: 102
There is no search number
Search number: -1
PS C:\ds\practice02\sol>
```