자료구조 보고서

Homework#12

학과 : 소프트웨어학과
학번 : 2016039040
이름 : 윤용진

1. Sorting & hashing에 관한 것이다.
    (a) hw12-sorting-hashing.c 컴파일하고 실행해 본다.
    (b) hw12-sorting-hashing.c 함수를 분석하여 동작과정을 이해한다.
    (c) 이해한 부분을 주석으로 남긴다.
    (d) 파일명은 hw12-sorting-hashing.c으로 자신의 GitHub에 코드를 업로드한다.

```
/*
 * hw12-sorting-hashing.c
 *
 *  Created on: May 22, 2021
 *
 *  Homework 12: Sorting & Hashing
 *  Department of Computer Science at Chungbuk National University
 *
 */

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

#define MAX_ARRAY_SIZE                  13      /* prime number */
#define MAX_HASH_TABLE_SIZE         MAX_ARRAY_SIZE

/* 필요에 따라 함수 추가 가능 */
int initialize(int **a);
int freeArray(int *a);
void printArray(int *a);

int selectionSort(int *a);
int insertionSort(int *a);
int bubbleSort(int *a);
int shellSort(int *a);
/* recursive function으로 구현 */
int quickSort(int *a, int n);


/* hash code generator, key % MAX_HASH_TABLE_SIZE */
int hashCode(int key);

/* array a에대 한 hash table을 만든다. */
```

```
int hashing(int *a, int **ht);

/* hash table에서 key를 찾아 hash table의 index return */
int search(int *ht, int key);


int main()
{
        char command;
        /* 정수 배열 생성 */
        int *array = NULL;
        /* 해쉬 테이블 생성 */
        int *hashtable = NULL;
        int key = -1;
        int index = -1;

        srand(time(NULL));

        do{

printf("------------------------------------------------------------------\n");
                printf("                                        Sorting & Hashing
      \n");

printf("------------------------------------------------------------------\n");
                printf(" Initialize      = z          Quit              = q\n");
                printf(" Selection Sort  = s          Insertion Sort    = i\n");
                printf(" Bubble Sort     = b          Shell Sort        = l\n");
                printf(" Quick Sort      = k          Print Array       = p\n");
                printf(" Hashing         = h          Search(for Hash) = e\n");

printf("------------------------------------------------------------------\n");

                printf("Command = ");
                scanf(" %c", &command);

                switch(command) {
                case 'z': case 'Z':
                        initialize(&array);
                        break;
```

```
                case 'q': case 'Q':
                        freeArray(array);
                        break;
                case 's': case 'S':
                        selectionSort(array);
                        break;
                case 'i': case 'I':
                        insertionSort(array);
                        break;
                case 'b': case 'B':
                        bubbleSort(array);
                        break;
                case 'l': case 'L':
                        shellSort(array);
                        break;
                case 'k': case 'K':
                        printf("Quick Sort: \n");

printf("-------------------------------------------------------------\n");
                        printArray(array);
                        quickSort(array, MAX_ARRAY_SIZE);

printf("-------------------------------------------------------------\n");
                        printArray(array);

                        break;

                case 'h': case 'H':
                        printf("Hashing: \n");

printf("-------------------------------------------------------------\n");
                        printArray(array);
                        hashing(array, &hashtable);
                        printArray(hashtable);
                        break;

                case 'e': case 'E':
                        printf("Your Key = ");
                        scanf("%d", &key);
                        printArray(hashtable);
```

```
                        index = search(hashtable, key);
                        printf("key = %d, index = %d,  hashtable[%d] = %d\n",
key, index, index, hashtable[index]);
                        break;

                case 'p': case 'P':
                        printArray(array);
                        break;
                default:
                        printf("\n        >>>>>   Concentration!!   <<<<<      \n");
                        break;
                }

        }while(command != 'q' && command != 'Q');

        return 1;
}

/* 초기화 */
int initialize(int** a)
{
        int *temp = NULL;

        /* array가 NULL인 경우 메모리 할당 */
        if(*a == NULL) {
                temp = (int*)malloc(sizeof(int) * MAX_ARRAY_SIZE);
                *a = temp;  /* 할당된 메모리의 주소를 복사 --> main에서 배열을
control 할수 있도록 함*/
        } else
                temp = *a;

        /* 랜덤값을 배열의 값으로 저장 */
        for(int i = 0; i < MAX_ARRAY_SIZE; i++)
                temp[i] = rand() % MAX_ARRAY_SIZE;

        return 0;
}

/* 배열 메모리 해제 */
int freeArray(int *a)
```

```
{
        if(a != NULL)
                free(a);
        return 0;
}

/* 배열 출력 */
void printArray(int *a)
{
        if (a == NULL) {
                printf("nothing to print.\n");
                return;
        }
        for(int i = 0; i < MAX_ARRAY_SIZE; i++) /* 배열 인덱스 출력 */
                printf("a[%02d] ", i);
        printf("\n");
        for(int i = 0; i < MAX_ARRAY_SIZE; i++) /* 배열 원소 값 출력 */
                printf("%5d ", a[i]);
        printf("\n");
}

/* 선택정렬 가장 작은 값을 찾아 앞으로 꺼낸다 */
int selectionSort(int *a)
{
        int min; /* 가장 작은 원소 */
        int minindex; /* 가장 작은 원소의 인덱스 */
        int i, j;

        printf("Selection Sort: \n");

printf("-------------------------------------------------------------\n");

        printArray(a); /* 정렬 전 배열 */

        for (i = 0; i < MAX_ARRAY_SIZE; i++)
        {
                minindex = i;
                min = a[i];
                /* 가장 작은 원소 탐색 */
                for(j = i+1; j < MAX_ARRAY_SIZE; j++)
```

```
                {
                        if (min > a[j])
                        {
                                min = a[j];
                                minindex = j;
                        }
                }
                /* 기준위치의 원소와 가장 작은 원소의 위치를 서로 바꾼다 */
                a[minindex] = a[i];
                a[i] = min;
        }


printf("----------------------------------------------------------------\n");
        printArray(a); /* 정렬 후 배열 */
        return 0;
}

/* 삽입 정렬 : 첫번째 원소와 가장 작은 원소를 1대1로 교환하는 선택 정렬과는 달리 가장 작은 원소를 맨 앞으로 꺼내오는 정렬 - 정렬된 앞부분과 정렬되지 않은 뒷부분으로 나뉜다(두개의 부분집합)*/
int insertionSort(int *a)
{
        int i, j, t;

        printf("Insertion Sort: \n");

printf("----------------------------------------------------------------\n");

        printArray(a); /* 정렬 전 배열 */

        for(i = 1; i < MAX_ARRAY_SIZE; i++) /* 배열의 2번째 원소부터 탐색 */
        {
                t = a[i]; /* 현재 탐색중인 i+1번째 원소 값을 임시 저장 */
                j = i;      /* 원소의 인덱스 임시 저장 */
                while (a[j-1] > t && j > 0) /* 왼쪽의 인접 원소가 첫번째 원소가 아니면서 임시저장한 원소보다 크다면 반복 */
                {
                        a[j] = a[j-1]; /* 왼쪽의 원소를 오른쪽으로 밀어낸다 */
                        j--; /* 좌측으로 탐색 */
```

```
                }
                a[j] = t; /* 왼쪽 원소가 t보다 작게 되는 위치에 원소를 삽입하여 정렬
*/
        }


printf("----------------------------------------------------------------\n");
        printArray(a); /* 정렬 후 배열 */

        return 0;
}

/* 버블 정렬 두 인접한 원소를 검사하여 정렬 - 최대값을 제일 뒤로 보내는 것을 반복
*/
int bubbleSort(int *a)
{
        int i, j, t;

        printf("Bubble Sort: \n");

printf("----------------------------------------------------------------\n");

        printArray(a); /* 정렬 전 배열 */

        for(i = 0; i < MAX_ARRAY_SIZE; i++)
        {
                for (j = 0; j < MAX_ARRAY_SIZE; j++) /* 최대갑을 갖는 원소를 맨
뒤로 보낸다 */
                {
                        /* 왼쪽의 원소 값이 오른쪽 원소 값보다 크다면 자리 교체 */
                        if (a[j-1] > a[j])
                        {
                                t = a[j-1]; /* t에 왼쪽 원소 값 임시 저장 */
                                a[j-1] = a[j]; /* 왼쪽 위치에 오른쪽 원소 값 저장 */
                                a[j] = t; /* 오른쪽 원소에 임시 저장한 왼쪽 원소 값
저장 */
                        }
                }
        }
```

```
printf("------------------------------------------------------------\n");
        printArray(a); /* 정렬 후 배열 */

        return 0;
}

/* 셸 정렬 역순 배열에 성능이 떨어지는 삽입 정렬을 보완 */
int shellSort(int *a)
{
        int i, j, k, h, v;

        printf("Shell Sort: \n");

printf("------------------------------------------------------------\n");

        printArray(a); /* 정렬 전 배열 */

        /* 일정한 간격 h를 통해 삽입 정렬 */
        for (h = MAX_ARRAY_SIZE/2; h > 0; h /= 2)
        {
                for (i = 0; i < h; i++)
                {
                        /* 인접한 두 원소가 아닌 간격 h만큼 떨어진 원소들을 비교하
여 삽입 정렬 */
                        for(j = i + h; j < MAX_ARRAY_SIZE; j += h)
                        {
                                v = a[j];
                                k = j;
                                while (k > h-1 && a[k-h] > v)
                                {
                                        a[k] = a[k-h];
                                        k -= h;
                                }
                                a[k] = v;
                        }
                }
        }

printf("------------------------------------------------------------\n");
```

```
	printArray(a); /* 정렬 후 배열 */

	return 0;
}

/* 퀵 정렬 기준 값(피봇) 을 중심으로 왼쪽과 오른쪽을 나누어 정렬 */
int quickSort(int *a, int n)
{
	int v, t;
	int i, j;

	if (n > 1)
	{
		v = a[n-1]; /* 오른쪽 끝 원소를 기준값으로 선택 */
		i = -1;
		j = n - 1;

		while(1)
		{
			while(a[++i] < v); /* 배열의 왼쪽부터 v보다 작거나 같은 원소를 탐색, while문 종료 후 i는 v보다 큰 원소의 인덱스 */
			while(a[--j] > v); /* 배열의 오른쪽부터 v보다 크거나 같은 원소를 탐색, while문 종료 후 j는 v보다 작은 원소의 인덱스 */

			if (i >= j) break; /* i가 j보다 크거나 같다면 반복문 탈출 */
			/* i번째 원소와 v번째 원소의 위치를 서로 교체 */
			t = a[i];
			a[i] = a[j];
			a[j] = t;
		}
		/* i번째 원소와 기준 값 원소의 위치를 서로 교체 */
		t = a[i];
		a[i] = a[n-1];
		a[n-1] = t;

		quickSort(a, i); /* i를 기준값 으로 왼쪽 부분집합을 퀵정렬 */
		quickSort(a+i+1, n-i-1); /* 오른쪽 부분집합을 퀵정렬 */
	}

	return 0;
```

```
}

/* 제산함수 */
int hashCode(int key) {
   return key % MAX_HASH_TABLE_SIZE; /* 홈 버킷 인덱스 리턴 */
}

int hashing(int *a, int **ht)
{
        int *hashtable = NULL;

        /* hash table이 NULL인 경우 메모리 할당 */
        if(*ht == NULL) {
                hashtable = (int*)malloc(sizeof(int) * MAX_ARRAY_SIZE);
                *ht = hashtable;  /* 할당된 메모리의 주소를 복사 --> main에서 배
열을 control 할수 있도록 함*/
        } else {
                hashtable = *ht;        /* hash table이 NULL이 아닌경우, table 재
활용, reset to -1 */
        }

        for(int i = 0; i < MAX_HASH_TABLE_SIZE; i++)
                hashtable[i] = -1;

        /*
        for(int i = 0; i < MAX_HASH_TABLE_SIZE; i++)
                printf("hashtable[%d] = %d\n", i, hashtable[i]);
        */

        int key = -1;
        int hashcode = -1;
        int index = -1;
        for (int i = 0; i < MAX_ARRAY_SIZE; i++)
        {
                key = a[i]; /* 원소 값을 키로 저장 */
                hashcode = hashCode(key); /* 키에 대한 해시코드 생성 */
                /*
                printf("key = %d, hashcode = %d, hashtable[%d]=%d\n", key,
hashcode, hashcode, hashtable[hashcode]);
                */
```

```
                /* 해시코드를 인덱스로하는 해시테이블이 비어있는 경우 키(데이터)를
저장 */
                if (hashtable[hashcode] == -1)
                {
                        hashtable[hashcode] = key;
                }
                /* 충돌이 일어나면 */
                else {
                        /* 비어있는 버킷을 탐색해 데이터 저장 */
                        index = hashcode;

                        while(hashtable[index] != -1)
                        {
                                index = (++index) % MAX_HASH_TABLE_SIZE;
                                /*
                                printf("index = %d\n", index);
                                */
                        }
                        hashtable[index] = key;
                }
        }

        return 0;
}

int search(int *ht, int key)
{
        /* 키 값에 대한 해시코드를 생성해 인덱스에 저장 */
        int index = hashCode(key);

        /* 인덱스를 통해 해시테이블 탐색*/

        /* 저장된 데이터가 키 값과 일치하면 인덱스를 리턴 */
        if(ht[index] == key)
                return index;

        /* 데이터가 키 값과 일치하지 않으면, 다른 버킷들을 탐색해 데이터를 찾는다
*/
        while(ht[++index] != key)
```

```
        {
                index = index % MAX_HASH_TABLE_SIZE;
        }
        return index;
}
```

4. GitHub에 hw12 Repository를 생성하고 hw12-sorting-hashing.c를 업로드 한다.

https://github.com/uniz21/DataStructure-HW-12

실행결과

```
[----- [Yoon Yong Jin] [2016039040] -----]
------------------------------------------------------------
                      Sorting & Hashing
------------------------------------------------------------
 Initialize       = z          Quit             = q
 Selection Sort   = s          Insertion Sort   = i
 Bubble Sort      = b          Shell Sort       = l
 Quick Sort       = k          Print Array      = p
 Hashing          = h          Search(for Hash) = e
------------------------------------------------------------
Command = z
------------------------------------------------------------
                      Sorting & Hashing
------------------------------------------------------------
 Initialize       = z          Quit             = q
 Selection Sort   = s          Insertion Sort   = i
 Bubble Sort      = b          Shell Sort       = l
 Quick Sort       = k          Print Array      = p
 Hashing          = h          Search(for Hash) = e
------------------------------------------------------------
Command = s
Selection Sort:
------------------------------------------------------------
a[00] a[01] a[02] a[03] a[04] a[05] a[06] a[07] a[08] a[09] a[10] a[11] a[12]
    1    11     5    10     0     1     3     2     4    12    10     2     9
------------------------------------------------------------
a[00] a[01] a[02] a[03] a[04] a[05] a[06] a[07] a[08] a[09] a[10] a[11] a[12]
    0     1     1     2     2     3     4     5     9    10    10    11    12
```

```
-----------------------------------------------------------------
                      Sorting & Hashing
-----------------------------------------------------------------
 Initialize       = z          Quit             = q
 Selection Sort   = s          Insertion Sort   = i
 Bubble Sort      = b          Shell Sort       = l
 Quick Sort       = k          Print Array      = p
 Hashing          = h          Search(for Hash) = e
-----------------------------------------------------------------
Command = z
-----------------------------------------------------------------
                      Sorting & Hashing
-----------------------------------------------------------------
 Initialize       = z          Quit             = q
 Selection Sort   = s          Insertion Sort   = i
 Bubble Sort      = b          Shell Sort       = l
 Quick Sort       = k          Print Array      = p
 Hashing          = h          Search(for Hash) = e
-----------------------------------------------------------------
Command = i
Insertion Sort:
-----------------------------------------------------------------
a[00] a[01] a[02] a[03] a[04] a[05] a[06] a[07] a[08] a[09] a[10] a[11] a[12]
    0     2     7    10     8    11     0     0     7     1     2     5     6
-----------------------------------------------------------------
a[00] a[01] a[02] a[03] a[04] a[05] a[06] a[07] a[08] a[09] a[10] a[11] a[12]
    0     0     0     1     2     2     5     6     7     7     8    10    11
-----------------------------------------------------------------
```

```
------------------------------------------------------------
                      Sorting & Hashing
------------------------------------------------------------
 Initialize       = z          Quit             = q
 Selection Sort   = s          Insertion Sort   = i
 Bubble Sort      = b          Shell Sort       = l
 Quick Sort       = k          Print Array      = p
 Hashing          = h          Search(for Hash) = e
------------------------------------------------------------
Command = z
------------------------------------------------------------
                      Sorting & Hashing
------------------------------------------------------------
 Initialize       = z          Quit             = q
 Selection Sort   = s          Insertion Sort   = i
 Bubble Sort      = b          Shell Sort       = l
 Quick Sort       = k          Print Array      = p
 Hashing          = h          Search(for Hash) = e
------------------------------------------------------------
Command = k
Quick Sort:
------------------------------------------------------------
a[00] a[01] a[02] a[03] a[04] a[05] a[06] a[07] a[08] a[09] a[10] a[11] a[12]
    0     9     4     0     8     6     0     5     1     1     7     5     5
------------------------------------------------------------
a[00] a[01] a[02] a[03] a[04] a[05] a[06] a[07] a[08] a[09] a[10] a[11] a[12]
    0     0     0     1     1     4     5     5     5     6     7     8     9
```

```
-----------------------------------------------------------------
                      Sorting & Hashing
-----------------------------------------------------------------
 Initialize       = z          Quit             = q
 Selection Sort   = s          Insertion Sort   = i
 Bubble Sort      = b          Shell Sort       = l
 Quick Sort       = k          Print Array      = p
 Hashing          = h          Search(for Hash) = e
-----------------------------------------------------------------
Command = z
-----------------------------------------------------------------
                      Sorting & Hashing
-----------------------------------------------------------------
 Initialize       = z          Quit             = q
 Selection Sort   = s          Insertion Sort   = i
 Bubble Sort      = b          Shell Sort       = l
 Quick Sort       = k          Print Array      = p
 Hashing          = h          Search(for Hash) = e
-----------------------------------------------------------------
Command = l
Shell Sort:
-----------------------------------------------------------------
a[00] a[01] a[02] a[03] a[04] a[05] a[06] a[07] a[08] a[09] a[10] a[11] a[12]
   12     9     7     3     6     6     1    12     1     5     5     1    10
-----------------------------------------------------------------
a[00] a[01] a[02] a[03] a[04] a[05] a[06] a[07] a[08] a[09] a[10] a[11] a[12]
    1     1     1     3     5     5     6     6     7     9    10    12    12
```

```
------------------------------------------------------------
                      Sorting & Hashing
------------------------------------------------------------
 Initialize       = z          Quit             = q
 Selection Sort   = s          Insertion Sort   = i
 Bubble Sort      = b          Shell Sort       = l
 Quick Sort       = k          Print Array      = p
 Hashing          = h          Search(for Hash) = e
------------------------------------------------------------
Command = z
------------------------------------------------------------
                      Sorting & Hashing
------------------------------------------------------------
 Initialize       = z          Quit             = q
 Selection Sort   = s          Insertion Sort   = i
 Bubble Sort      = b          Shell Sort       = l
 Quick Sort       = k          Print Array      = p
 Hashing          = h          Search(for Hash) = e
------------------------------------------------------------
Command = h
Hashing:
------------------------------------------------------------
a[00] a[01] a[02] a[03] a[04] a[05] a[06] a[07] a[08] a[09] a[10] a[11] a[12]
    6     3     2     4     9     1     4     0     6     2    12     9     7
a[00] a[01] a[02] a[03] a[04] a[05] a[06] a[07] a[08] a[09] a[10] a[11] a[12]
    0     1     2     3     4     4     6     6     2     9     9     7    12
------------------------------------------------------------
                      Sorting & Hashing
------------------------------------------------------------
 Initialize       = z          Quit             = q
 Selection Sort   = s          Insertion Sort   = i
 Bubble Sort      = b          Shell Sort       = l
 Quick Sort       = k          Print Array      = p
 Hashing          = h          Search(for Hash) = e
------------------------------------------------------------
Command = e
Your Key = 7
a[00] a[01] a[02] a[03] a[04] a[05] a[06] a[07] a[08] a[09] a[10] a[11] a[12]
    0     1     2     3     4     4     6     6     2     9     9     7    12
key = 7, index = 11,  hashtable[11] = 7
```

```
------------------------------------------------------------------
                      Sorting & Hashing
------------------------------------------------------------------
 Initialize       = z          Quit            = q
 Selection Sort   = s          Insertion Sort  = i
 Bubble Sort      = b          Shell Sort      = l
 Quick Sort       = k          Print Array     = p
 Hashing          = h          Search(for Hash) = e
------------------------------------------------------------------
Command = z
------------------------------------------------------------------
                      Sorting & Hashing
------------------------------------------------------------------
 Initialize       = z          Quit            = q
 Selection Sort   = s          Insertion Sort  = i
 Bubble Sort      = b          Shell Sort      = l
 Quick Sort       = k          Print Array     = p
 Hashing          = h          Search(for Hash) = e
------------------------------------------------------------------
Command = b
Bubble Sort:
------------------------------------------------------------------
a[00] a[01] a[02] a[03] a[04] a[05] a[06] a[07] a[08] a[09] a[10] a[11] a[12]
    1    12     4    12     4     2     4     2     4     8     0     5     5
------------------------------------------------------------------
a[00] a[01] a[02] a[03] a[04] a[05] a[06] a[07] a[08] a[09] a[10] a[11] a[12]
    1     1     2     2     4     4     4     4     5     5     8    12    12
------------------------------------------------------------------
```