자료구조 (Data Structure)

Programming Assignment 2

이름: Jumagul Alua

학번: 20231632

과목: CSE2080-02

HW2

문제 1. Recursive Power Set

이번 문제는 C++에서 작성했다. 하지만 출력이 순서대로 안 나와서 문제이다.

아래와 같이 3 개의 문자 입력을 받는다고 했을때, P(S)를 모두 출력하게 될 것이다.

```
Enter the size: 3
{} {c} {b} {b c} {a} {a c} {a b} {a b c}
```

코드의 몇가지 부분을 살표보자.

입력을 숫자로 받기 때문에 아래와 같은 알파벳 문자를 저장하는 작업을 해주었다.

```
char set[26];
    for (int i = 0; i < setSize; ++i) {
        set[i] = 'a' + i;
    }
```

PowerSet 함수에는,

현재 원소를 부분집합에 포함하거나 포함하지 않는 두 가지 경우로 나누어 계속해서 재귀 호출을 한다. 모든 원소에 대해 이런 과정을 반복하여 모든 가능한 부분집합을 생성하여 출력된다.

문제 2. string 과 pattern 이 matching 되는 pmatch_all 함수

- 수업 때 배운 pmatch 함수의 declarations 을 사용하였다:

#include #include #define max_string_size 100

#define max_pattern_size 100 int pmatch();

void fail(); int failure[max_pattern_size];

char string[max_string_size]; char pat[max_pattern_size];

다만, string size 의 max 값을 31 로 설정한다.

“void pmatch_all(char *string, char *pat)” 이 함수에는:

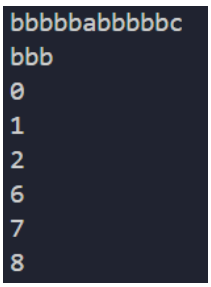
- 검색할 대상이 될 요소를 저장할 string 포인터이며, 검색할 패턴을 입력 받을 pat 이 있다.
- failure 라는 길이가 max_pattern_size 인 배열을 선언하여 이 배열은 패턴에 대한 실패 함수를 저장하기 위해 사용되었다.
- 다음엔, i 와 j 를 초기화하고, 문자열과 패턴을 비교하기 시작한다.
- 문자열과 패턴을 한 문자씩 비교하면서 일치하는 경우에는 i 와 j 를 함께 증가시킨다. 일치하지 않는 경우에는 패턴을 이동시킨다.

```
void pmatch_all(char *string, char *pat) {
    int failure[max_pattern_size];
    int lens = strlen(string);
    int lenp = strlen(pat);

    fail(pat, failure);

    int i = 0, j = 0;
    while (i < lens) {
        if (string[i] == pat[j]) {
            if (j == lenp - 1) {
                printf("%d\n", i - j);
                j = failure[j];
            }
            i++;
            j++;
        }
```

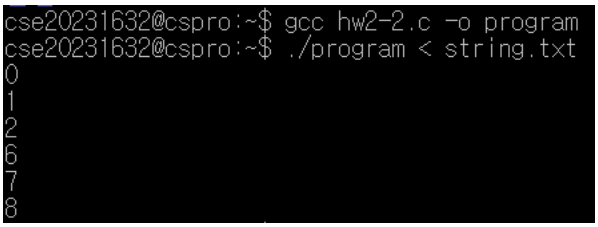
```
        else if (j > 0)
            j = failure[j - 1] + 1;
        else
            i++;
    }
}
```

- ‘void fail’ 은 패턴에 대한 실패 함수를 계산하는 함수이다.

출력: 이런 식으로 입력을 받아 발견된 인덱스를 출력한다.



또한, gcc 를 이용해 컴파일 할 수 있다. ‘String.txt’ 파일에다가 입력할 요소를 추가하고, 아래와 같이 출력을 받는다.



문제 3. check_array()

- void Sorting() 함수는 배열을 정렬하는데 사용된다. 배열의 요소를 하나씩 확인하는 Insertion Sort 알고리즘이 구현되었다.
- void check_array(): 이 함수는 입력 받은 배열이 연속된 숫자로만 이루어져 있는지 확인한다. 먼저 Sorting 함수를 호출하여 배열을 정렬하고, 정렬된 배열을 둘러보면서 각

요소가 다음 요소와 1 만큼 차이가 나는지 확인한다. 만약, 차이가 나지 않는다면 연속된 숫자가 아니므로 0 을 반환하고, 모든 요소가 연속된 숫자라면 1 을 반환한다.

- main 함수: 이 함수는 배열의 크기인 n 을 입력 받은 후에, malloc 한다. 그런 다음, 사용자로부터 n 개의 정수를 입력 받는다. 그리고 check_array 함수를 호출하여 배열이 연속된 숫자로만 이루어져 있는지 확인하고 결과를 출력한다.
- 전체 코드의 시간 복잡도는 O(n)이다.

출력 예시:

```
5
2 3 1 5 4
1
```

```
6
3 5 8 1 0 3
0
```

문제 4. Sort function

제약 조건이 있어, sorting 함수 직접 구현했으며, string.h 사용 불가능해서 compare 함수를 구현하였다. Student.txt file 에다가 학생의 수와 학생들의 이름을 적혀서 main 함수에는 파일을 열어준다. 학생들의 성과 이름을 비교해서 Lexical order 에 따라 정렬을 해봤다.

그리고 student.txt 을 불러서 컴파일을 해 정렬된 출력을 얻었다.

```
cse20231632@cspro:~$ gcc hw2-4.c -o pop
cse20231632@cspro:~$ ./pop < student.txt
Cho Yujin
Choi Hojeong
Choi Minjeong
Kim Minju
Kim Minsu
Lee Minsu
```