과제 2 보고서

20231609 정희선

문제 1

모든 subsets을 recursive하게 출력하는 문제이다.

1. 사용하는 data structure
- char array(char *set) : set S의 모든 element를 저장한다.
- string array(char **subsets) : set의 모든 subset을 저장하는 데 사용된다.
- temporary array(char *current) : 생성 중인 subset을 임시로 저장하는 데 사용된다.

2. algorithm 설명
main 함수의 코드 흐름에 따라 algorithm을 설명할 것이다.

- set의 크기 입력: set S의 크기 n을 입력 받는다.
- data structure 초기화: 입력된 크기에 따라 `set`, `current`, `subsets`를 초기화한다. set에는 a부터 시작하는 n개의 char이 저장된다.
- subset 생성: PowerSet 함수를 호출하여 모든 subset을 생성한다. 이 함수는 recursive하게 subset을 구성하여 subsets에 저장한다.
- subset 정렬: qsort 함수를 사용하여 subsets array의 subset을 길이에 따라 오름차순으로 정렬한다.
- subset 출력: subset을 순서대로 출력한다. subset은 printSubset 함수를 통해 출력된다.
- memory 정리: 동적으로 할당된 memory(`set`, `current`, `subsets` 및 각 subset)를 해제한다.

문제 2

file에서 읽은 text와 pattern에서 KMP algorithm을 통해 text에서 patter의 모든 출현 위치를 찾는 문제이다.

1. 사용하는 data structure
- char array(char *txt , char *pat): 각각 text와 pattern을 저장하는데 사용된다.
- int array(int *lps): pattern에서 각 index에 대해 접두사와 접미사가 일치하는 최대 길이를 저장한다.

2. algorithm 설명
main 함수의 코드 흐름에 따라 algorithm을 설명할 것이다.

- 입력 인자 검사: 인자가 올바르지 않은 경우 error messege를 출력하고 종료한다.
- 파일 열기 : 입력받은 파일명을 사용하여 파일을 연다.
- text와 pattern 읽기: 파일에서 text와 pattern을 읽어 각각 `txt`와 `pat` array에 저장합니다.
- pattern matching 실행: `pmatch_all` 함수를 호출하여 text 내에서 pattern의 모든 출현 위치를 찾는다. 이 때 KMP(Knuth-Morris-Pratt) algorithm을 사용한다.
  - LPS array 계산: `computeLPSArray` 함수를 사용하여 pattern array에 대한 lps array을 계산한다.
  - text scan: text를 scan면서 pattern과의 일치를 검사한다. pattern이 일치하면 해당 위치를 출력하고, 다음 비교 위치를 조정한다.
- 자원 정리: 파일을 닫고 프로그램을 종료한다.

문제 3

배열이 연속된 정수의 범위를 포함하는지 확인하는 문제이다.

1. 사용하는 data structure
Int array(int arr[]) : 입력 받은 숫자들을 저장하는데 사용된다. 며, 이 숫자들은 array에 연속적인 정수들이 있는지 확인하는 데 사용된다.

2. algorithm 설명
main 함수의 코드 흐름에 따라 algorithm을 설명할 것이다.

- array 크기 입력 : array 크기 n을 입력 받는다.
- array 초기화: 입력 받은 크기의 array `arr`을 생성하고, array의 각 요소를 입력 받는다.
- array 검사: `check_array` 함수를 호출하여 입력받은 array가 연속적인 정수들로 구성되어 있는지 확인한다.
  - check_array 함수에서는 array내의 `min_val`과 `max_val`을 찾고, array 요소들의 `sum`을 계산한다.
  - 연속적인 정수의 경우, `max_val - min_val == n - 1` 이어야 하고, `sum`이 min_val에서 max_val까지 연속적인 정수들의 합(`expected_sum`)과 일치해야 한다.
- 결과 출력: `check_array` 함수의 반환 값에 따라, array가 연속적인 정수들로만 구성되어 있는지 확인된 결과를 출력한다. 반환 값이 `1`이면 배열이 연속적인 정수들로 구성되어 있음을, `0`이면 그렇지 않음을 의미한다.

문제 4

입력받은 이름들을 memory에 저장하고, 이름들을 사전 순으로 정렬한 후 출력하는 문제이다.

1. 사용하는 data structure
- char pointer array(char **names): 동적으로 memory를 할당받아 사용자로부터 입력받은 이름들을 저장한다.

2. algorithm 설명
main 함수의 코드 흐름에 따라 algorithm을 설명할 것이다.

- 입력 크기 읽기: 몇 개의 이름을 입력받을 것인지 크기 `n`을 입력 받는다.
- memory allocation: `n`개의 pointer를 저장할 수 있는 `names` array에 대한 memory를 할당한다.
- 이름 입력 받기: 각 이름에 대해 100 문자까지 저장할 수 있는 공간을 동적으로 할당하고, 표준 입력으로부터 이름을 읽어 배열에 저장한다.
- 이름 sorting: `sort_names` 함수를 사용하여 이름들을 사전 순으로 정렬한다. 이 함수 내에서 bubble sorting algorithm을 사용하며, `strcmp_custom` 함수를 통해 문자열을 비교한다.
- sorted 이름 출력: 정렬된 이름들을 출력한다.
- 자원 정리: 각 이름에 할당된 메모리와 `names` 배열 자체에 할당된 메모리를 해제한다.