**Algorithms [Assignment #3]: Implementation of 0-1 Knapsack Problem**

**주제 및 목표**: 0-1 knapsack 문제에서 최적의 solution을 얻는 알고리즘들을 구현하고 평가함

**내용**:

* **Backtracking 기법을 활용한 0-1 knapsack 문제 해결**
* **Branch-and-bound 기법을 활용한 0-1 knapsack 문제 해결**

**보고서**:

* 1. 개요, 2. 구현 언어 및 방법, 3. 실험 결과 및 분석, 4. 결론 형태로 기술   
  (표지 포함 7p이하)
* **Branch-and-bound** 기법은 **breadth- 및 best-first 탐색 기법을 모두 포함**
* **각 기법들 (i.e., depth-, breadth-, and best-first)의 평가 및 논의 포함 필수**

**데이터 및 구현 언어 (language)**:

* 제공된 knapsack list를 활용해서 구현함 (input.txt)..
* C 혹은 C++ 언어 사용
  + STL 사용 불가 (직접 memory 할당 및 자료구조 개발)

**평가 방법**:

* 분석 **보고서** (50%), **source code** (50%)

**제출 기한: 12월 03일 (금) 23:59**

**문제 정의:**

* 개의 물건에는 무게 kg와 가치 가 측정되어 있다. 무게 kg까지 버틸 수 있는 가방에 물건들의 가치 총 합이 최대로 되도록 넣으려고 한다. 넣을 수 있는 물건들의 가치의 총 합을 출력하면 된다.

**입력 범위 제한:**

* 모든 입력은 정수이다.

**입력:**

* input.txt 텍스트 파일로 주어진다.
* 첫째 줄에 과 가 공백으로 구분되어 주어진다.
* 둘째 줄에는 번째 물건의 무게 가 공백으로 주어진다.
* 셋째 줄에는 번째 물건의 가치 가 공백으로 주어진다.

**출력:**

* 가방에 담을 수 있는 물건들의 가치의 합의 최대를 출력한다.

**제출 프로그램 명세:**

* + 구동 형식
    - 명령어 : ./실행파일 [실행 option] < [입력 파일]
    - 예시 : ./test.exe best < input.txt
    - [입력 파일]: 제공된 input.txt 파일
    - [실행 option]: 실행 option ( [“backtracking”, “breadth”, “best”] )
    - 비주얼 스튜디오를 사용하면 “명령 인수”를 이용하여 사용 가능
  + 실행 option
    - 1. backtracking : Bracktracking 기법을 이용하여 계산된 결과를 출력
    - 2. breadth : Branch-and-bound 중 breadth 탐색 기법을 이용하여 계산된 결과를 출력
    - 3. best : Branch-and-bound 중 best-first 탐색 기법을 이용하여 계산된 결과를 출력
* 표준입출력(stdin, stdout)을 사용할 것, **(주의! 파일 입출력과 다름)**
* 본 프로그램은 반드시 GCC 환경에서 컴파일 및 실행이 되어야 함.

**제출 양식 명세:**

* 아래 파일을 zip 파일로 압축하여 제출 ( ex. 압축 파일 명 : 학번\_이름\_분반\_과제3.zip )
  + 보고서 PDF 파일 형식 ( ex. 파일명 : 학번\_이름\_분반\_과제3.pdf )
  + 소스코드 .c 또는 .cpp 파일 형식 ( ex. 학번.c 또는 학번.cpp )

**유의사항 :**

* GCC 컴파일 환경에서 컴파일 안된다면 감점 또는 0점 처리
* 보고서, 소스코드 등 Copy & Paste는 0점 처리
* 제출 양식 반드시 지킬 것.
* 기간 내에 제출할 것.
* 불필요한 출력은 하지 말 것.
* **소스코드가 정상 작동하는 부분은 채점 프로그램을 이용하기 때문에 출력 형식에 맞도록 할 것(다른 출력은 제출 시 주석처리 필수), GCC 컴파일 환경에서 채점을 하기 때문에 더더욱 유의**