2020 Spring OOP Assignment Report

과제 번호 : ASSN1
학번 : 20190439
이름 : 오승훈
Povis ID : sho0927

명예서약 (Honor Code)

나는 이 프로그래밍 과제를 다른 사람의 부적절한 도움 없이 완수하였습니다.
I completed this programming task without the improper help of others.

프로그램을 하다 보면 결정해야 할 세부 사항이 많은데, 이러한 세부 사항을 처리한 방법과 이유를 보고서에 쓰십시오.
독창적인 아이디어와 추가 기능은 보너스 점수를 받을 수 있으므로, 보고서에 명확히 기재하십시오.
제출 기한이 지나면 LMS를 통하여 과제를 제출할 수 없습니다. 제출 기한이 지난 후에는 담당 조교에게 email로 보내야 합니다. 늦게 제출된 과제에 대한 감점은 담당 조교에게 email이 도착한 시간을 기준으로 합니다.
문제가 여러 개인 경우, 각 문제별로 정리해서 작성합니다.

각 문항별 설명은 편의를 위한 것으로, 삭제하고 제출한다.

1. 프로그램 개요

a. ASSN1-1

A. ASSN1-1의 경우 가로 세로의 정사각형의 개수가 주어지고, 그 정사각형의 대각선으로 선을 그을 때 이 대각선으로 인해서 나눠지지 않은 사각형의 개수를 구하는 방법에 대한 어싸인이다.

B. 이 것을 구하는 방법은 힌트에 나와있듯이 최대공약수를 구해서 문제를 푼다.

b. ASSN1-2

A. ASSN1-2의 경우 학생 수가 주어지고 그 학생들은 1~n까지의 번호가 붙어져있을 때, 학생들 중 리코더를 잃어버린 친구들이 있고, 추가적인 리코더를 가지고 있는 친구들이 있다. 이 때 추가적인 리코더를 가지고 있는 친구의 경우에는 자신의 앞 번호 또는 뒷 번호 친구에게 리코더를 줄 수 있을 때 최대 몇 명이 리코더를 가지고 수업에 갈 수 있는지 구하는 문제이다.

B. 이 경우 본인의 것을 잃어버리지 않고, 추가적인 리코더를 가지고 있는 경우 친구가 0개의 리코더를 가지고 있을 때 빌려줄 수 있다고 생각을 하면 되고, 학생 수 만큼 array를 선언한 후, 리코더를 잃어버리기만한 친구를 0, 리코더를 잃어버렸지만 추가적인 리코더를 가지고 있는 친구 또는 잃어버리지도 추가적인 리코더도 가지고 있지 않았던 친구는 1로, 잃어버리지 않았고, 추가적인 리코더를 가지고 있는 친구를 2라고 생각을 해서 문제를 푼다.

c. ASSN1-3

A. 이 문제의 경우 위스키가 있고, 위스키가 순서대로 있을 때 최대 마실 수 있는 양을 구하는 문제이다.

B. 이 문제의 경우 dynamic programming 을 활용하는 문제로 수학적인 방식으로 문제를 풀게 되면 점화식을 이용해서 문제를 푸는 것이다.

2. 프로그램의 구조 및 알고리즘

a. ASSN 1-1

A. Width, height 는 가로 세로를 뜻하는 변수이며, number은 최대공약수를 의미하는 변수이다.

B. 이번 ASSN1-1에서는 최대공약수를 찾는 것이 관건이었다. 그래서 최대공약수를 찾기 위해서 두 수를 1부터 작은수까지 나눠보면서 나머지가 0인 수를 n에 저장해서 가장 큰 수를 number에 저장할 수 있도록 했다.

C. 그리고 온전한 사각형의 개수를 width * height – (width + height – number)이 되는 데 그 이유는 최대 공약수만큼 사각틀이 있을 때 내부에서 교점이 생기게 되어있다. 그렇기 때문에 교점이 생길 때 마다 원래 나눠져야 하는 사각형이 3개에서 2개로 줄기 때문에 최대 공약수만큼 width, height에서 빼줘야하는 것이었다.

b. ASSN1-2

A. Student의 경우 학생수를 뜻하고, l_student는 lost_student 즉 리코더를 잃어버린 학생의 수, a_student는 additional_student 즉 여분의 리코더를 가지고 있는 학생의 수 , number는 잃어버린 학생과 여분의 리코더를 가지고 있는 학생의 번호를 일시적으로 저장하기 위한 변수이며, count 는 마지막 총 리코더를 가지고 있을 수 있는 학생의 수를 저장하기 위한 변수이다. 또한 student_ary는 학생들이 리코더를 가지고 있는 개수를 저장하기 위한 배열이 된다.

B. 이 문제의 경우에는 array를 학생의 수만큼의 공간을 할당을 해서 문제를 해결했는데 리코더를 잃어버리기만한 친구를 0, 리코더를 잃어버렸지만 추가적인 리코더를 가지고 있는 친구 또는 잃어버리지도 추가적인 리코더도 가지고 있지 않았던 친구는 1로, 잃어버리지 않았고, 추가적인 리코더를 가지고 있는 친구를 2라고 생각을 해서 문제를 푼다.

C. 첫 번째 학생은 뒤의 학생에게만 마지막 학생의 경우에는 앞의 학생에게만 여분의 리코더가 있을 때 리코더를 빌려 줄 수 있기 때문에 그 경우를 예외 처리해서 본인이 2일 때 줄 수 있는 학생이 0인 경우 각각의 학생이 1 1이 되는 array로 바꿔주며, 나머지 학생의 경우 앞 뒤 모두 줄 수 있는데, 0 2 0인 경우 자신보다 번호가 앞인 학생에게 먼저 빌려줄 수 있도록 알고리즘을 구성했다. 그 이유는 만약 1 0 2 0 2 인 경우 뒤의 학생에게 먼저 빌려줄 수 있도록 알고리즘을 구성한다면 1 0 1 1 2가 되어 4명의 학생이 쓸 수 있지만 앞의 학생에게 먼저 준다면 1 1 1 1 1이 되므로 5명의 학생이 쓸 수 있게 되기 때문이다.

D. 그 후 마지막에 array에 리코더 개수가 1 또는 2개인 학생의 수를 세서 출력을 한다.

c. ASSN1-3

A. N의 경우 위스키의 잔 수를 받아오는 변수이며, input은 각 잔에 채워진 양을 넣어줄 array이며, dynamic은 각 경우에서 마실 수 있는 최대의 양을 넣어줄 array가 된다.

B. Input array에 일단 1부터 n번째까지의 위스키의 양을 받아온다.

C. 그 후 dynamic에는 각 경우에 마실 수 있는 최대의 양을 쓰는 것인데 dynamic[1]의 경우 첫 번째 잔을 마시는 경우가 최대의 경우가 되는 것이고, dynamic[2]의 경우 첫 번째, 두 번째 모두 마시는 경우가 최대가 되는 경우일 것이다. 하지만 세 번째 경우 세 잔 연속으로 마실 수 없는 조건 때문에 최대 값은 이 세 가지를 비교해봐야할 것이다. 첫 번째 잔과 두 번째 잔을 마신 경우, 두 번째 잔과 세 번째 잔을 마신 경우, 첫 번째 잔과 세 번째 잔을 마신 경우 중 가장 큰 값을 골라야 할 것이다. 하지만 이 경우를 생각해보면 첫 번째 잔과 세 번째 잔을 마신 경우는 dynamic[1]과 input[3]을 마신 경우이고, 첫 번째 잔과 두 번째 잔을 마신 경우는 dynamic[2]인 경우이고, 두 번째 잔과 세 번째 잔을 마신 경우에는 dynamic[0]에 input[2]와 input[3]을 더한 경우이다. 즉 이것을 일반화하면 이 세 가지 중 가장 큰 값을 고르는 알고리즘을 만들어내면 되는 것이다. 여기서 dynamic[0]이 필요한 이유는 점화식의 일반화를 위해서이다.

3. 토론 및 개선

- 이번 과제를 통해서 가장 아쉬웠던 점은 ASSN1-3에서 내가 생각했던 방법은 linked list를 이용하고, structure을 이용하여 현재 마셨는지 안 마셨는지, 전에 마셨는지 안 마셨는지, 전전에 마셨는지 안 마셨는지와 현재 마신 값을 저장하는 structure을 만들어서 linked list를 이용하여 연결을 해두고, 모든 경우의 수를 고려하되 현재, 전, 전전이 모두 같은 경우 amount를 비교하여 더 작은 값은 연결을 끊어서 총 7개 13개 7개 13개의 경우가 계속 반복되서 최종적으로 값을 구해내는 알고리즘을 구현해 내려고 하였으나 모든 것을 고쳐봐도 output 값이 정상적으로 나오지 않아서 결국 구글링을 하게 되었고, 이 문제를 점화식으로 풀 수 있다는 것을 알게 되었다. 내가 생각한 방법을 실제 코드로 한 번 더 옮겨서 문제를 풀어보고 싶었으나 시간이 없다는 아쉬움이 너무 컸던 것 같다.

- 또한 이번 ASSN에서 오히려 가장 오랜 시간을 차지한 부분은 어싸인을 구현하는 부분이 아니라 makefile을 하는 과정에서 너무 오랜 시간을 차지했다는 부분이 아쉬움으로 남는 것 같다. Makefile을 하는 취지는 이해가 되나 makefile을 구글링을 하면 쉽게 하는 방법이 나와있을 것이라고 해서 makefile을 찾아는 보았으나 이 과정이 정확히 어떤 과정이고, mingw에서 이 과정이 필요한 이유, makefile 소스코드에서 이 코드들이 어떤 것들을 의미하는지 더 정확히 pdf 파일로 정리를 해서 이해를 할 수 있었으면 더 좋았을 것 같다.

4. 참고 문헌

- https://mygumi.tistory.com/98
- https://en.wikipedia.org/wiki/Greedy_algorithm
- https://jhb.kr/53
- https://bowbowbow.tistory.com/12
- https://github.com/dmlc/xgboost/issues/1132
- http://doc.kldp.org/KoreanDoc/html/GNU-Make/GNU-Make-4.html
- https://maeuminpaper.tistory.com/92
- https://snowdeer.github.io/c++/2017/09/06/how-to-use-make-utility/
- https://www.tuwlab.com/ece/27193