5

경시대회 문제의 교육적 활용

국민대학교 ▌최 준 수*
한국외국어대학교 ▌신 찬 수*

1. 서 론

국내외의 초, 중, 고, 대학생 대상의 프로그래밍 경시대회에서는 게임이나 퍼즐 같은 간단한 문제에서부터 수학적 논리와 다양한 알고리즘의 기법을 활용해서 해결해야 하는 복합적인 문제에 이르기까지 많은 종류의 문제가 상, 중, 하의 난이도에 따라 출제된다. 따라서 자료구조나 알고리즘 강의에서 수업 진도에 맞게 경시대회 문제를 선택하여 경시대회와 유사한 방식으로 출제/채점/평가한다면, 수강 학생들 간의 경쟁을 통한 동기 유발 뿐 만 아니라 추상적인 알고리즘을 구체적인 프로그램 코드로 구현해보는 실전 경험을 제공할 수 있다. 다양한 난이도의 2000여개 이상의 문제가 공개되어 쉽게 얻을 수 있고, 온라인으로 제출된 학생들의 프로그램을 자동으로 채점하여 점수를 관리해주는 웹 시스템도 잘 갖추어져 있어 자료구조나 알고리즘 강의에 활용하기 위한 좋은 조건이 이미 마련되어 있다.

본고에서는 이러한 환경에서 경시대회 문제를 실제 강의(자료구조/알고리즘/컴퓨터프로그래밍)에 활용하는 방법 및 활용 예를 구체적으로 살펴본다. 우선 국내외의 유명 프로그래밍 경시대회의 종류 및 특징을 (2절에서) 살펴보고, 경시대회 문제를 자료구조나 알고리즘 강의에 활용하는 방법과 실제 활용 예를 (3절에서) 살펴본다. 또한 C/C++/Java와 같은 컴퓨터 프로그래밍 언어 강의에서의 활용방안을 구체적인 문제 사례를 중심으로 (4절에서) 살펴본다. 마지막으로 보다 효율적인 활용을 위해 보완해야할 몇 가지 점을 살펴보고 결론을 맺는다.

2. 프로그래밍 경시대회

세계적인 프로그래밍 경시대회에는 대학생을 대상으로 하는 ACM 국제 대학생 프로그래밍 경시대회(ACM-ICPC, ACM-International Collegiate Programming Contest)[1]와 중고등학생을 대상으로 하는 국제 정보 올림피아드(IOI, International Olympiad in Informatics)[2,3], 그리고 일반인을 포함한 모든 프로그래머를 대상으로 하는 상업적인 경시대회인 탑코더(TopCoder) 대회[4] 등이 있다.

ACM-ICPC는 같은 대학의 학생 3명으로 구성된 팀 단위의 경시대회이면서, 대학별 경시대회이다. 전 세계를 북미, 남미, 유럽, 아프리카, 아시아, 오세아니아 등의 6개 대륙으로 나누고, 대륙을 또 다시 세분화한 수 십개의 지역으로 나누어서, 각 지역의 대학생을 대상으로 하는 지역대회를 개최하고, 그 지역의 1등 입상자들을 모아서 최종 결선 대회(World Final)를 치루는 형식으로 진행된다. 대회는 일반적으로 5시간의 제한시간 내에 주어진 5~10개의 문제를 가장 많이 해결하는 팀이 우승하는 형식으로 진행된다. 각 팀에서 제출된 문제 해결 프로그램은 경시대회 현장에서 실시간으로 채점되며, 채점 결과에 따라서 각 팀은 프로그램을 수정 보완하여 재 채점을 요청할 수 있다. 채점 시스템의 검증을 통과한 문제만 만점으로 인정한다. 각 문제에 대하여 부분 점수는 없으며, 정답을 만들어 내는 프로그램을 제출하는데 걸리는 시간이 짧을수록 더 높은 점수를 받게 된다. 또한 재 채점에 따른 시간감점이라는 요소를 도입되어 있어 같은 문제를 해결한 여러 팀이 있는 경우에는 감점을 적게 받은 팀이 더 좋은 성적을 받도록 하고 있다. 따라서 각 팀이 높은 점수를 받기 위해서는 해답 프로그램을 빠르게 만들어 내는 것도 중요하지만, 완벽한 프로그램이 되도록 세심하게 코딩할 필요가 있다.

IOI는 중고등학생을 위한 프로그래밍 경시대회로서, 올림픽과 유사한 국가별 경시대회이다. 각 참가국에서는 네 명의 대표학생들이 대회에 참여하며, 학생 팀 단위의 경시대회인 ICPC와는 달리 개개인의 학생이 혼자서 모든 문제를 해결해야하는 개인별 경시대

* 종신회원

회이다. 대회는 이틀에 걸쳐 열리게 되며, 일반적으로 하루에 3문제가 출제되고, 이 문제들을 5시간 내에 해결하여야 한다. IOI 대회에서는 채점은 모든 경시대회가 종료된 이후에 배치작업에 의하여 채점이 이루어지고, 정답을 구한 테스트 데이터의 개수에 비례한 부분 점수가 주어진다. 주어진 문제에 대한 정확한 프로그램을 작성하였는지의 여부를 테스트 데이터에 의존하게 되므로, 테스트 데이터는 매우 신중하게 만들어지게 되며, 매우 다양한 경우를 고려한 데이터를 작성하게 된다[5].

탑코더 경시대회는 상금이 걸려있는 일종의 프로 프로그래밍 경시대회로서, 이 경시대회를 통하여 수준 높은 프로그래머를 발굴하는 것이 하나의 목적이다. 이 경시대회에 참가한 프로그래머는 경시대회 결과를 추후에 유망한 기업의 입사 자료로도 활용할 수 있다. 기업에서는 탑코더 주관회사에 지원한 프로그래머의 대회 결과를 조회할 수 있을 뿐만 아니라, 이 프로그래머가 작성한 프로그램을 직접 조회할 수 있도록 하여 보다 정확한 프로그램의 수준을 판별할 수 있는 정보를 제공하고 있다.

ACM-ICPC, IOI, 탑코더 대회를 포함한 많은 경시대회에 사용된 테스트 데이터는 공개되지 않는다. 이러한 대회에 참여하는 참가자들을 위한 가장 중요한 권고사항 중의 하나는, 대충 작동하는 프로그램을 작성하는 것이 아니라, 많은 테스트 케이스에 대하여 모두 정답을 구하는 올바른 프로그래밍을 하기 위해서는 주어진 문제를 정밀하게 분석해야 할 뿐만 아니라 주어진 조건을 세밀히 검토하여 프로그램에 반영하여야 한다는 것이다. 또한 입력되는 형식에 맞게 데이터를 입력할 뿐만 아니라, 형식에 맞게 출력하는 것도 매우 중요하다.

ACM-ICPC 대회의 지역예선 혹은 최종 결선대회, IOI, 탑코더 경시대회에 출제되는 문제는 매우 수준 높은 문제로서 깊은 수학적인 지식이나 알고리즘 설계에 관한 지식을 필요로 하여, 일반 대학생들이 시도하기에는 너무 어려운 문제일 수 있다. 비록 IOI대회가 중고등학생을 대상으로 하는 경시대회이지만 문제의 난이도는 대학생을 대상으로 하는 ICPC 대회와 동등한 수준의 고난이도 문제가 출제된다.

이들 세계적인 대회보다 수준이 낮은 문제가 출제되는 대회로서는 ACM-ICPC의 지역대회 예선에 출전하기 위하여 각 대학에서 개최하는 대학별 예선 경시대회가 있으며, IOI 대회에 출전할 국가대표 학생을 선발하기 위하여 개최되는 국가별 경시대회가 있다[3]. 예를 들어, 우리나라에서는 IOI 대회에 출전할 예비 학생들을 한국 정보올림피아드 대회(KOI, Korea Olympiad in Informatics)[6]를 통하여 매년 20여명씩 선발하고 있다.

이러한, ICPC 지역예선에 출전할 대학별 예선대회나 IOI, 국가대표를 선발할 국가내의 예선대회, 혹은 미국의 많은 대학 혹은 고등학교에서 주최하는 프로그래밍 경시대회에 출제되는 문제의 난이도는 세계적인 본선 대회 문제의 난이도 보다는 상대적으로 쉬운 문제이지만, 특별한 훈련이 없는 대학생들에게는 어려운 문제일 수도 있다.

3. 경시대회 문제의 자료구조 및 알고리즘 강의에의 활용

ACM-ICPC, IOI와 같은 경시대회에 출제되는 문제는 크게 수학적인 문제, 게임 혹은 퍼즐 류의 문제, 알고리즘 설계 기법을 사용해서 해결할 수 있는 문제 등으로 분류된다. 이러한 문제들은 일반적으로 문제 출제자들이 딱딱한 이론적인 문제를 설명하기 쉽고, 이해하기 쉬우며, 재미있는 문제의 형식으로 변환하여 제출한다[5]. 따라서 표면적으로는 재미있는 퀴즈 혹은 퍼즐과 같은 문제들이지만 내부적으로는 깊은 수학이나 알고리즘의 이론을 이용해야만 해결할 수 있는 문제들이 대부분이다.

ACM-ICPC, IOI에서 출제된 많은 문제들은 스페인의 바야돌리드 대학교(Univerdidad de Valladolid)에서 관장하고 있는 온라인 자동채점 시스템[7]에 수록되어 있다. 이 시스템에서는 현재 약 2,000여개의 기 출제된 문제가 수록되어 있으며, 모든 문제에 대하여 온라인으로 재출되는 문제 해결 프로그램을 채점해 주고 있다. 이 시스템은 주로 ACM-ICPC 대회에 참가하고자 하는 전 세계의 많은 대학생들이 대회를 준비하기 위한 훈련 시스템으로 주로 사용하고 있으며, IOI에 참가하고자 하는 많은 중고등학생들도 사용하고 있다. 이 시스템에서는 현재까지 약 550만개 정도의 프로그램을 채점하였다고 알려져 있다.

스토니 브룩 소재 뉴욕 주립대의 Skiena 교수와 바야돌리드 대학의 Revilla 교수는 바야돌리드 대학의 자동채점 시스템에서 관리하고 있는 문제들 중에서 대학교육에 활용할 수 있는 가장 좋은 문제들을 선택하여, 이 문제들을 온라인에서 자동채점해 주는 시스템을 가동하고 있다[8,9]. 이 시스템에서는 바야돌리드 대학에서 관리하고 있는 문제들을 여러 주제별로 나누고, 이 주제에 맞는 다양한 문제들을 선택하여

제공하고 있다. 예를 들어 수학적인 기반의 문제들로는 대수학, 정수론, 조합론, 그리고 기하학적인 도형을 처리하는 분야의 문제들이 있다. 또한 기초적인 자료구조를 이용하는 문제들도 제공되어 있으며, 알고리즘 분야의 문제들로는 정렬을 기반으로 하는 문제들과, 동적 프로그래밍(dynamic programming), 그래프 탐색을 이용하는 문제, 그래프 알고리즘에 관련된 문제, 백트래킹(backtracking)을 이용하는 문제, 계산기하학(computational geometry)에 관련된 문제들이 있다.

이러한 문제들은 ACM-ICPC, IOI 경시대회를 준비하는 학생들을 훈련하기 위한 강의 자료로 활용될 수 있을 뿐만 아니라, 자료구조 및 알고리즘 강의에서 프로그래밍 실습을 위한 자료로 활용될 수 있으며, 프로그래밍 강의에서 알고리즘을 다루는 자료로 활용될 수 있다. 특히, 알고리즘 강의에서 추상적으로 설명되는 이론적인 알고리즘의 설계 부분을 구체적으로 구현할 수 있는 자료로 활용하면 알고리즘 강의의 훌륭한 부교재로서 사용될 수 있다.

보다 효과적인 교육적 효과를 위해선, 여러 가지의 자료구조와 알고리즘 기법을 종합적으로 훈련할 수 있는 복합적 문제를 발굴하여 수업에 활용하는 것이다. 가장 대표적인 예로, 이차원 미로 탈출 문제가 있다. 이 문제는 다양한 형태로 변형되어 IOI나 ACM-ICPC 문제로 출제되기도 한다. 미로 탈출 문제는 미로의 모양이나 장애물의 종류 및 통과 규칙 등에 의해 다양하게 변형될 수 있으며, 탈출 경로의 길이나 탈출 시간 등의 여러 가지 목적함수로 알고리즘의 성능을 측정할 수 있다는 장점이 있다. 미로 탈출 문제를 해결하는 과정을 통해, 2차원 배열과 스택 자료구조 사용을 연습하게 되며, 미로를 그래프 모델로 정의하여 그래프에서의 DFS/BFS 탐색 방법 및 최단거리 계산 알고리즘의 원리도 학습할 수 있다. 또한 장애물의 통과 규칙에 따라, 백트랙킹(backtracking) 알고리즘 기법의 원리도 이해하게 된다. 현재 한국외국어대학교 디지털정보공학과에선 올해 2학기부터 알고리즘 과목 수강생을 대상으로 미로탈출경진대회를 수업의 설계과제로 포함시킬 계획이다. 담당 교수는 학생들의 미로탈출 알고리즘을 수행하여 그 결과를 실시간으로 화면에 그려주는 사용자 인터페이스를 작성하여 미리 제공한다. 학생은 이 인터페이스를 이용하여 미로를 생성하고, DLL 형식으로 저장한 자신의 탈출 알고리즘 코드(C/C++)를 인터페이스와 결합하여 알고리즘 수행 결과를 확인하고 이를 알고리즘 개선작업에 이용하게 된다. 경진대회는 토너먼트 형식으로 진행하며, 하나의 미로를 탈출하는 과정을 실시간으로 대회 참가자들에게 보여주어 컴퓨터 게임 대회에 익숙한 학생들의 호기심을 자극하고 적극적인 참여를 유도한다. 이러한 복합적 알고리즘 문제의 "게임화"는 학생들의 학업 성취도를 높일 수 있는 가장 효과적인 방법으로 생각되며, 더욱 다양한 문제 발굴이 필요하다고 판단된다.

또한, 강의에서 더 쉽게 이 시스템을 활용할 수 있도록 하기 위하여, 강의 담당자가 이 시스템에서 강의를 개설할 수 있도록 하고 있다. 개설된 강의에서 과제로 출제될 문제를 선택해 놓으면, 강의에 등록된 학생들이 문제를 이 시스템에 직접 과제물을 제출하게 하여, 강사들에게는 각 문제들의 채점 및 학생들의 총점수를 통합하여 관리하는 기능을 제공해 준다. 또한, 각 학생들이 제출한 프로그램 코드를 확인할 수 있도록 해주며, 학생들의 프로그램이 표절하였는지도 확인 가능하다.

4. 경시대회 문제의 컴퓨터 C 프로그래밍 강의에의 활용

4.1 실습 문제

ACM-ICPC, IOI 경시대회에 출제되는 문제는 깊이 있는 수학적인 지식과 알고리즘 설계 지식이 없는 경우에는 그 해결방법을 단시간에 구하기 매우 어려운 문제들로 구성되어 있다. 따라서 이러한 경시대회의 문제는 알고리즘을 다루는 과목에서 실제로 알고리

표 1

수열에서 연속적인 세 자연수의 관계
자연수의 수열이 주어졌을 때, 이 수열에서 연속적인 세 개 자연수 a b c의 관계가 다음과 같은 경우가 각각 몇 가지인지를 계산하는 프로그램을 작성하시오. 1. 세 자연수가 오름차순으로 정렬되어 있는 경우, 즉, $a \leq b \leq c$ 인 경우 2. 세 자연수가 내림차순으로 정렬되어 있는 경우, 즉, $a \geq b \geq c$ 인 경우 3. 세 자연수가 모두 같은 경우 4. 세 자연수 중에서 두 개의 정수는 같고, 한 정수는 다른 경우 5. 세 자연수가 모두 다른 경우

예를 들어, 다음과 같은 10개로 만들어진 정수의 수열에서

4 2 7 7 7 6 9 9 5 2

세 자연수가 오름차순으로 정렬되어 있는 경우는 (2, 7, 7), (7, 7, 7), (6, 9, 9)의 세 가지가 있으며, 세 자연수가 내림차순으로 정렬되어 있는 경우는(7, 7, 7), (7, 7, 6), (9, 9, 5), (9, 5, 2)의 네 가지가 있으며, 세 자연수가 모두 같은 경우는 (7, 7, 7)의 한 가지가 있으며, 세 자연수 중에서 두 개의 정수는 같고, 한 정수가 다른 경우는 (2, 7, 7), (7, 7, 6), (6, 9, 9), (9, 9, 5)의 네 가지가 있고, 세 자연수가 모두 다른 경우는 (4, 2, 7), (7, 6, 9), (9, 5, 2)의 세 가지가 있다.

입력

입력 파일의 이름은 "input.txt" 이다. 입력은 t 개의 테스트 케이스로 주어진다. 입력 파일의 첫 번째 줄에 테스트 케이스의 개수를 나타내는 정수 t 가 주어진다. 두 번째 줄부터 한 개의 테스트 케이스에 대하여 두 줄씩 데이터가 입력된다. 각 테스트 케이스에 해당하는 첫 번째 줄에는 먼저 수열에 나타나는 자연수의 개수를 나타내는 정수 n ($3 \leq n \leq 100$) 이 주어진다. 두 번째 줄에는 수열에 나타나는 자연수 n 개가 한 줄에 주어진다. 이 자연수들은 100 보다 작거나 같다. 모든 자연수들 사이에는 한 개의 공백이 있으며, 잘못된 데이터가 입력되는 경우는 없다.

출력

출력은 표준출력(standard output)을 사용한다. 입력되는 테스트 케이스의 순서대로 다음 줄에 이어서 각 테스트 케이스의 결과를 출력한다. 각 테스트 케이스에 해당하는 출력의 첫 줄에 연속적인 세 자연수의 관계가 위 1, 2, 3, 4, 5의 경우에 해당되는 경우의 수를 순서대로 출력한다. 각 정수들 사이에는 한 개의 공백을 둔다.

입력의 예	출력의 예
3	3 4 1 4 3
10	8 8 8 0 0
4 2 7 7 7 6 9 9 5 2	50 0 0 5
10	
1 1 1 1 1 1 1 1 1 1	
7	
1 2 4 8 16 32 64	

즘을 구현해보는 실습 도구로서 활용할 수 있다. 그러나 경시대회에 출제되는 문제보다는 상대적으로 쉬우면서 직관적으로 문제해결 방법을 구할 수 있는 문제와 이 문제의 해결 프로그램을 자동으로 채점할 수 있는 시스템을 구축하게 되면 이 시스템을 기초 프로그래밍 강의에 활용할 수 있다.

대부분의 컴퓨터관련 학과에서는 C, C++ 혹은 Java 등의 프로그래밍 언어 중에서 학생들에게 처음으로 프로그래밍 언어를 접할 수 있도록 1학년 혹은 2학년의 저학년에서 프로그래밍 관련과목을 개설하고 있다. 이러한 과목에서는 각 프로그래밍 언어의 기능, 구문 및 배열과 같은 간단한 자료구조를 배우게 된다. 이러한 강의에서 학습한 프로그래밍 언어를 활용하여 해결할 수 있는 실습 문제로서는 매우 단순한 문제인 경우로 제한적이며, 특히 자료구조 강의를 수강하기 이전 단계이므로 스택(stack), 큐(queue), 연결 리스트(linked list)와 같은 가장 간단한 자료구조를 이용하는 문제조차도 사용할 수 없으며, 가장 복잡한 자료구조가 2차원 배열 정도이다. 이러한 제한적인 범주에 속하는 문제들로서는 ICPC 세계대회 문제뿐 만 아니라 지역예선에서 출제되는 조금 복잡한 문제는 사용될 수 없으며, 지역예선 대회에서의 가장 간단한 문제 정도가 사용될 수 있다. 이러한 문제를 특별한 프로그래밍 언어 교육을 위한 문제로 변형하거나, 제약 조건을 가하게되면, 훌륭한 프로그래밍 언어의 실습 문제로 변형할 수 있다.

국민대학교 컴퓨터 공학부에서는 이러한 문제들과 문제들을 해결하는 프로그램의 자동 채점 시스템을 구축하여, 1학년 2학기에 개설된 C 프로그래밍 언어나 2학년 1학기에 개설된 C++프로그래밍 언어 강의에서 과제물의 자동 채점 시스템으로 활용하고 있다.

예를 들어, 표 1은 C 프로그래밍 언어에서 if-문을 포함하는 조건문, 관계연산자(relational operators), 논리연산자(logical operators)를 실습할 수 있는 문제로서, 상대적으로 난이도가 낮은 문제이지만, 정확한 관계연산자와 논리연산자를 구사할 수 있도록 학습하는 좋은 문제의 예이다.

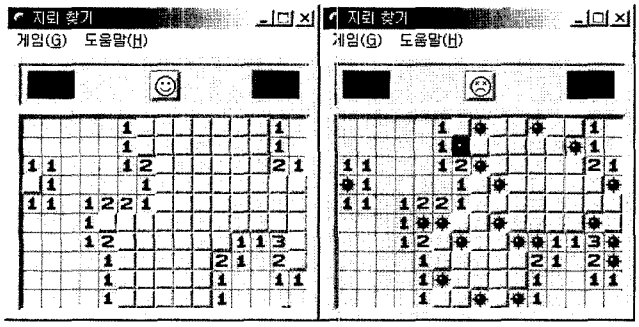
다음은 2차원 배열을 활용하여 해결하는 문제의 예이다(표 2 참조).

주어진 문제를 해결하는 방법에는 여러 가지가 있을

표 2

지뢰찾기 게임

간단한 게임 프로그램 중에서 인기 있는 게임 중의 하나는 "지뢰 찾기" 게임이다. 이 게임은 크기가 $m \times n$ 인 바둑판 모양의 게임판에서 이루어진다. 이 게임판의 작은 정사각형 상자(셀이라 부름)에는 지뢰가 한 개 숨겨져 있을 수 있으며, 게임판에는 여러 개의 지뢰가 숨겨져 있다. 게임은 이 정사각형 셀을 클릭하면서 진행이 되고, 셀을 클릭하면 그 셀이 열리게 된다. 이 때, 그 셀에 지뢰가 숨겨져 있으면, 지뢰가 터지고 게임이 끝나게 된다. 만약 지뢰가 숨겨져 있지 않은 셀을 열게 되면, 그 셀에 인접한 여덟 개의 셀 중에서 지뢰가 있는 셀이 1개 이상인 경우에는 그 클릭한 셀만 열리게 되며, 그 셀 주위에 있는 지뢰의 개수를 클릭한 셀의 사각형 안에 표시해준다. 또한 인접한 여덟 개의 모든 셀에 지뢰가 없는 경우에는, 연쇄적으로 인접한 모든 여덟 개의 셀을 자동적으로 열게 된다. 이때 열리게 되는 셀은 앞에서와 같이 인접한 주위의 셀에 지뢰가 있는 경우에는 그 셀만 열리며, 인접한 주위의 셀에 지뢰가 없는 경우에는, 주위의 모든 셀을 다시 반복적으로 열게 된다.
아래 그림은 10×15인 게임판에 총 20개의 지뢰가 숨겨져 있는 게임이 진행되고 있는 것을 나타내고 있다. 왼쪽 그림에서는 지뢰가 터지지 않고 계속 진행된 상태이고, 오른쪽 그림은 지뢰가 숨겨진 셀을 열어서 게임이 끝난 상태를 나타낸다.



위와 같은 방법으로 클릭한 셀을 여는 경우에, 여러 개의 지뢰가 없는 임의의 셀을 열었을 때, 열리게 되는 모든 셀을 계산하는 프로그램을 작성하시오.

입력

입력 파일의 이름은 "input.txt"이다. 입력은 t개의 테스트 케이스로 주어진다. 입력 파일의 첫 번째 줄에 테스트 케이스의 개수를 나타내는 정수 t가 주어진다. 각 테스트 케이스의 첫 번째 줄에 두 개의 정수 m n ($2 \le m, n \le 100$) 이 주어진다. 두 정수 사이에는 하나의 공백이 있다. 여기서 m은 게임판의 가로의 길이를 나타내고, n은 세로의 길이를 나타내며, 단위 길이는 셀의 개수를 나타낸다. 두 번째 줄부터 n개의 줄에는 게임판에서 하나의 가로줄에 해당하는 셀에 지뢰가 있는지를 나타내는 m 개의 문자로 만들어진 스트링이 입력된다. 이 스트링에서 현재 지뢰가 있는 셀은 문자 '*'로 나타나며, 지뢰가 없는 셀은 문자 '+'로 표시된다. 또한, 사용자가 클릭하여 열어 본 지뢰가 없는 셀은 문자 '?'로 표시되어 있다. 클릭하여 열어 본 셀의 개수는 한 개 이상이다. 단, 입력되는 사용자가 열어 본 셀 중에는 다른 셀을 열어볼 때 자동적으로 열릴 수도 있다. 같은 줄에 입력되는 각 문자들 사이에는 공백이 없으며, 잘못된 데이터가 입력되는 경우는 없다.

출력

출력은 표준출력(standard output)을 사용한다. 입력되는 테스트 케이스의 순서대로 다음 줄에 이어서 각 테스트 케이스의 결과를 출력한다. 각 티스트 케이스에 해당하는 출력의 첫 줄부터 n 개의 각 줄에는 m 개의 문자를 출력하는데, 각 셀의 가로, 세로 위치에 출력되는 이 문자들의 종류에는 '0', '1', …, '8', 혹은 '+'가 있다. 문자 '0', '1', …, '8'은 각각 그 셀 주위에 있는 지뢰의 개수를 나타내는 정수 0, 1, …, 8을 나타내고, 문자 '+'는 아직 열리지 않은 셀을 나타낸다. 각 문자들 사이에는 공백을 두지 않는다.

입력의 예

```
2
15 10
??+++++*++*+++?
++++++*+++++*++
+++++++*+++++++
*+++++++*+++++*
++++++++++++++++
++++**++*++++*+
++++++*++**+++*
++++++++++++++*
+++++*+++++++++
?++++++*+*-+++?
```

출력의 예

```
000001+++++++10
000001+++++++10
1100012++++++21
+100001++++++++
1101221++++++++
0001++++++++++
00012++++++113+
00001+++++2102+
00001+++++10011
00001+++++10000
```

수 있으므로, 현재 강의하고 있는 내용과 관련된 방법으로 문제를 해결하도록 강제조항을 문제에 삽입하여 문제를 출제하는 경우도 있다. 예를 들어, 세 개의 정수를 정렬하는 문제는 버블정렬 등과 같은 간단한 정렬 알고리즘을 알고 있는 학생에게는 매우 쉬운 문제일 수 있다. 그러나 이 문제를 해결하기 위하여 배열을 사용하지 않으며 단순한 if-문을 사용하여 작성하도록 제약을 가하는 경우에는 if-문을 교육하는 좋은 문제가 될 수 있다. 또한 자동 채점 시스템에서는 제출된 프로그램이 배열을 사용한 일반 정렬 알고리즘을 사용한 경우에는 소스 코드 내에서 배열을 사용하였는지를 판별할 수 있는 필터링 작업을 통하여 배열을 사용하지 못하도록 할 수 있다(표 3 참조).

또한, 문제를 해결하기 위한 조건을 문제에서 제시

표 3

2-11의 배수

어떤 자연수 n이 2, 3, 4, …, 11의 배수인지를 검사하는 방법은 다음과 같다.

- 2의 배수 : n의 마지막 자리수가 2로 나누어지는 경우에는 2의 배수이다.
- 3의 배수 : n의 모든 자리수의 합이 3의 배수인 경우 경우에는 3의 배수이다.
- 4의 배수 : n의 마지막 두 자리수가 4로 나누어지는 경우에는 4의 배수이다.
- 5의 배수 : n의 마지막 자리수가 0이거나 5인 경우에는 5의 배수이다.
- 6의 배수 : n이 2와 3의 배수인 경우에는 6의 배수이다.
- 7의 배수 : 아래에서 설명
- 8의 배수 : n의 마지막 세 자리수가 8로 나누어지는 경우에는 8의 배수이다.
- 9의 배수 : n의 모든 자리수의 합이 9의 배수인 경우에는 9의 배수이다.
- 10의 배수 : n의 마지막 자리수가 0인 경우 경우에는 10의 배수이다.
- 11의 배수 : 홀수 번째 자리수의 합과 짝수 번째 자리수의 합의 차이가 11로 나누어지는 경우에는 11의 배수이다.

어떤 자연수가 7의 배수인지를 검사하기 위해서는 다음과 같은 방법을 사용한다. 맨 앞의 두 자리수에서 앞의 자리수에 3을 곱하여 뒤의 자리수에 더한 다음, 이 수를 맨 앞의 두 자리수를 대신하여 새로운 수를 만든다. 예를 들어, 39916800이 주어졌을 때, 맨 앞의 두 자리수 39에서 앞의 자리수 3에 3을 곱한 후 뒤의 자리수 9에 더하게 되면 3*3+9=18이 된다. 이렇게 만들어진 18을 맨 앞의 두 자리수 39를 대신하면 새로운 수 18916800가 만들어진다. 이 새로 만들어진 수에 대하여 앞에서 설명한 방법을 계속 적용하여, 한 자리수가 만들어질 때까지 반복적으로 적용한다. 최종적으로 만들어진 한 자리수가 7이면, 처음 주어진 수는 7의 배수이고, 그렇지 않은 경우는 7의 배수가 아니다.

입력

입력 파일의 이름은 "input.txt"이다. 입력은 t개의 테스트 케이스로 주어진다. 입력 파일의 첫 번째 줄에 테스트 케이스의 개수를 나타내는 정수 t가 주어진다. 두 번째 줄부터 t개의 줄에는 한 줄에 한 개의 테스트 케이스에 해당하는 매우 큰 자연수가 주어진다. 이 자연수는 최소 한 자리수이며, 최대 100자리수이다. 입력되는 자연수의 제일 높은 자리수의 숫자는 0이 아니며, 잘못된 데이터가 입력되는 경우는 없다.

출력

출력은 표준출력(standard output)을 사용한다. 입력되는 테스트 케이스의 순서대로 다음 줄에 이어서 각 테스트 케이스의 결과를 출력한다. 각 테스트 케이스에 해당하는 출력의 첫 줄에 입력되는 자연수가 2, 3, …, 11의 배수인지를 나타내는 결과를 순서대로 출력한다. 입력되는 수가 각 수(2, 3, …, 11)의 배수인 경우에는 1을 출력하고, 그렇지 않는 경우는 0을 출력한다. 각 정수들 사이에는 한 개의 공백을 둔다.

입력의 예

```
5
39916800
1
4712954379
34871903
47129543794712954379471295437947129543794712954379
```

출력의 예

```
1 1 1 1 1 1 1 1 1 1 1
0 0 0 0 0 0 0 0 0 0 0
0 1 0 0 0 1 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 1
0 1 0 0 0 1 0 0 0 0 0
```

하는 것 이외에도, 문제를 해결하기 위한 기본 틀을 제공하여서, 학습하고자 하는 특정 주제에 맞는 프로그램을 유도할 수 있다. 예를 들어, 다음과 같은 문제를 해결하기 위해서는 여러 방법이 있을 수 있으나, C 프로그래밍 언어의 함수 포인터, 함수 포인터의 배열을 학습하고자 하는 경우에는 미리 어느 정도 작성된 코드를 학생들에게 제공하고, 이를 기반으로 프로그램을 완성하도록 할 수 있다.

아래 표 4는 문제를 해결하도록 제시된 프로그램의 예로서 밑줄 쳐진 부분에 함수 포인터와 함수 포인터 배열을 사용하여 프로그램을 완성하도록 한다.

4.2 온라인 프로그래밍 과제물 자동채점 시스템

강의에서 활용되는 과제물 자동채점 시스템에서는 제출된 프로그램을 채점하는데 사용되는 테스트 데이터를 공개하지 않으며, 또한 제출된 프로그램이 완벽하게 모든 테스트 데이터에 대해서 정답을 구하지 못하는 경우에는 어떤 테스트 데이터에 대해서 정답을 구하지 못했다는 정보조차도 공개하지 않는다. 이는 프로그래머에게 주어진 문제를 다시 한 번 더 읽어 보게 함으로써, 보다 정확하게 문제를 분석하도록 하게 하고, 또한 주어진 문제의 조건에 맞도록 프로그래밍 되어 있는지를 재확인하도록 하는 훈련을 하도록 해 준다.

과제물 채점 시스템에서는 오답을 만들어 내는 프로그램에 대하여 부분 점수는 받을 수 없도록 하였다. 경시대회에서는 오답을 제출하는 경우에는 시간 벌점을 부과하는 경우도 있지만, 과제물 채점 시스템에서는 이러한 벌점제 없이 학생들은 무제한으로 반복하여 프로그램을 제출할 수 있도록 하였다. 정답을 과제

표 4

```
int isMultipleOf2(char *longInt);
int isMultipleOf3(char *longInt);
.....
int isMultipleOf11(char *longInt);

typedef int___pt2FunctionIsMultipleOfK_____;
pt2FunctionIsMultipleOfK isMultipleOfK[12] = {_____, _____, _____, _____, _____, _____,
        _____, _____, _____, _____, _____, _____};

void main(void)
{
    FILE *inFile;
    int i, k;
    int numCases;

    inFile = fopen("input.txt", "r");
    if(inFile == NULL) exit(1);

    fscanf(inFile, "%d", &numCases);

    for(i=0; i<numCases; i++)
    {
        char longInteger[101];

        fscanf(inFile, "%s", longInteger);

        for(k=2; k<=11; k++)
            printf("%d ", isMultipleOfK_____);

        printf("₩n");
    }

    fclose(inFile);
}

/* 주어진 수가 k ( 2 <= k <= 11 ) 의 배수인지를 검사하는 함수들 */
int isMultipleOf2(char *longInt)
{
    .....
}
```

물 기한 내에 제출하는 학생에 대해서는 모두 같은 점수를 받게 된다. 그러나 정답을 만들어내는 학생을 제출시간 순서대로 공개함으로서 학생들에게 먼저 과제물을 끝낸 노력에 대한 보상을 하도록 하였다. 또한 토론방을 만들어서 지금 프로그래밍하고 있는 문제에 대한 토론이나, 프로그래밍 디버깅이 필요한 경우에는 다른 학생들이나 담당 교수의 도움을 받을 수 있도록 하였다.

컴퓨터 프로그래밍 과제를 제출하는 강의에서 가장 큰 문제점이면서 해결하기 어려운 점은 학생들 간에 프로그램의 소스를 표절하는 것이다. 프로그램의 소스의 표절은 같은 강의를 수강하는 학생들 사이에서 다른 학생의 프로그램을 복사하거나 수정하는 것 이외에도, 인터넷에서 유포되고 있는 문제 해결 프로그램을 그대로 복사하여 제출하거나, 과제물에 맞게 수정하여 제출하는 경우를 포함하고 있다. 이러한 프로그램 소스의 표절을 탐지할 수 있는 방법[10]에 관한 연구는 많이 진행되고 있으나, 이에 대한 완벽한 방법은 아직 개발되고 있지 않다.

프로그래밍 과제가 가져오는 또 다른 문제 중의 하나는 학생들이 제출한 모든 프로그램에 대하여 프로그램이 문제에서 제시된 내용을 정확하게 수행하는 지를 파악하는 것 이외에도 소프트웨어 개발 관점에서 제출된 프로그램의 품질을 평가하기 어렵다는 점이다. 또한 C/C++/Java 프로그래밍에 강의에서 제출되는 과제물은 현재 강의에서 진행하고 있는 프로그래밍 언어의 기능을 실습하기 위한 과제물이 많으며, 이 과제물의 프로그램 소스의 길이는 짧게는 몇 십 라인에서, 평균적으로 몇 백 라인 이내인, 비교적 간단한 과제물인 경우가 많다. 이러한 짧은 프로그램에 대하여 소프트웨어 개발관점에서 평가하기가 쉽지 않고, 또한 이러한 프로그래밍 강의가 고학년에서 강의될 소프트웨어 개발관련 강의 이전에 학습하는 과정이다.

저 학년 학생들의 프로그래밍 개발 능력은 스스로 깨우치는 과정 이외에도 잘 작성된 프로그램을 분석하는 과정에서도 많이 개발되곤 한다. 실제로 과제물을 제출한 이후에도 많은 학생들은 다른 학생들이 이 과제물을 어떤 방법으로 또한 어떤 프로그램을 통하여 해결하였는지에 대해 매우 궁금해 하며, 다른 학생들의 프로그램을 보고자하는 욕구가 매우 강하다.

자동 채점 시스템에서는 제출된 프로그램이 문제를 정확하게 해결하는 프로그램인지만을 판별하며, 그 프로그램이 어느 정도로 소프트웨어 공학적인 측면에서, 잘 설계되어 있으며, 코딩되어 있는지에 대해서는 전혀 판별하지 않는다.

채점과정을 완전히 통과한 학생들의 정보는 실시간으로 공개하여, 현재 몇 명의 학생이 주어진 과제를 수행하였는지를 알 수 있도록 하는 것 뿐 만 아니라, 그 제출 순서도 공개하여서, 학생들의 경쟁심과 학업 성과를 제고하도록 하였다. 또한 자동 채점 시스템에서는 제출 기한이 지난 과제물에 대해서는 모든 소스 코드를 공개하고 있다. 소스 코드는 학생들이 제출하여 테스트 과정을 통과한 시간 순서대로 공개하고 있으며, 공개된 소스 코드를 다른 학생들이 조회한 횟수도 공개하여, 어느 학생의 코드들 가장 많이 조회하였는지를 알 수 있도록 하였다. 이렇게 소스 코드를 공개하는 장점으로는 다른 학생들이 작성한 코드를 검토하고, 자신의 프로그램과 비교함으로써, 같은 문제를 서로 다른 다양한 방법으로 해결하는 방법을 스스로 배우게 하는 것이다. 또 다른 부수적인 효과는 자신의 프로그램이 공개되어 누구나 조회할 수 있기 때문에, 다른 프로그램을 표절하는 등의 부정행위를 방지함에도 있다.

5. 결 론

본고에서는 국내외 경시대회 문제를 현재 전산관련 학과의 필수 교과목으로 운영 중인 컴퓨터언어/자료구조/알고리즘 강의에 활용 가능성과 실제 활용 예를 살펴보았다. 보다 많은 학교의 보다 다양한 강의에서 적극적으로 활용되기 위해선, 각 학교/학과의 상황에 적합한 문제관리시스템 및 자동채점시스템을 위한 표준 스펙을 공동연구를 통해 정하는 것이 필요하다고 판단된다.

참고문헌

[1] The ACM-ICPC International Collegiate Programming Contest, http://icpc.baylor.edu/icpc.

[2] International Olympiad in Informatics, http://www.ioinformatics.org.

[3] IOI = International Olympiad in Informatics, http://olympiads.win.tue.nl/ioi.

[4] TopCoder, http://www.topcoder.com.

[5] 임형석, ACM-ICPC 문제의 출제 및 채점 과정, 한국정보과학회지, 제25권, 제7호, 2007.

[6] 한국 정보올림피아드 대회(KOI, Korea Olympiad in Informatics), http://www.kado.or.kr/koi.

[7] On-Line Judge: Problem Set Archive, http://online-judge.uva.es.

[8] Welcome to Programming Challenges, http://www.programming-challenges.com.

[9] S. Skiena, M. Revilla, Programming Challenges: Then Programming Contest Training Manual, Springer, 2003.

[10] M. Wise, "YAP3: Improved Detection of Similarities in Computer Program and Other Texts", SIGCSEB: SIGCSE Bulletin (ACM Special Interest Group on Computer Science Education), Vol. 28, pp. 130-134, 1996.

최 준 수

1984 서울대학교 전기공학과(학사)
1986 한국과학기술원 전산학과(석사)
1995 뉴욕대학교 전산학과(박사)
1986~1996 한국통신 연구개발원(선임연구원)
1996~현재 국민대학교 컴퓨터공학부(교수)
관심분야 : 알고리즘, 계산기하학
E-mail : jschoi@kookmin.ac.kr

신 찬 수

1991 서울대학교 계산통계학과(학사)
1993 한국과학기술원 전산학과(석사)
1998 한국과학기술원 전산학과(박사)
1998~2001 한국과학기술원 전산학과 BK21 연구 교수
1999~2000 홍콩과학기술대(HKUST) 전산학과 Post Doc.
2001~현재 한국외국어대학교 전자정보공학부 (부교수)
관심분야 : 알고리즘, 계산기하학, 컴퓨터그래픽스
E-mail : cssin@hufs.ac.kr