Programming Language

Assignment #1

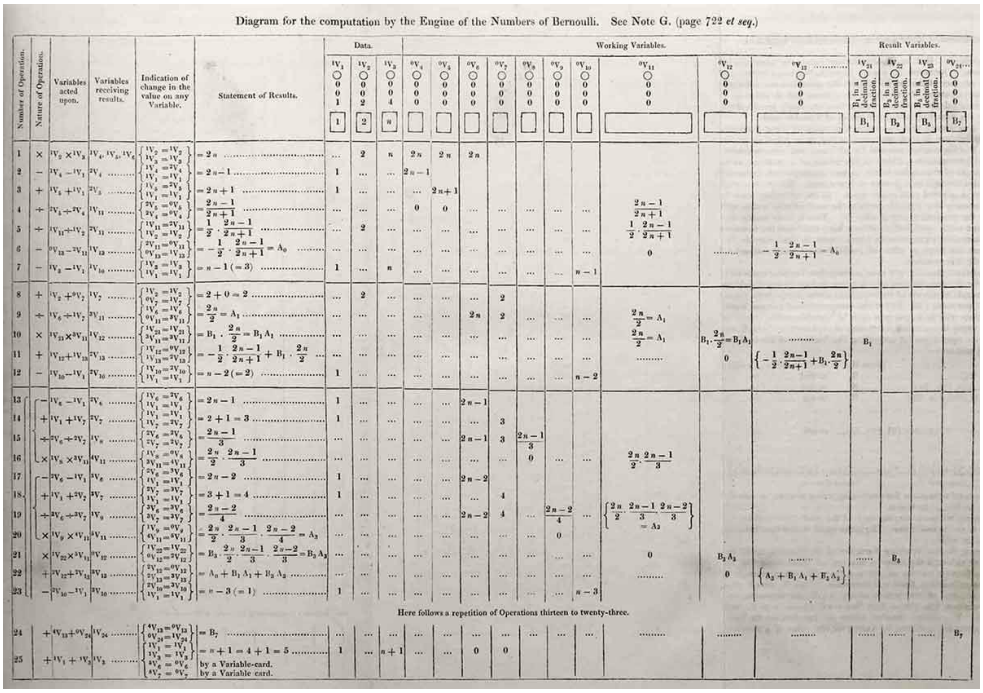
- **0 Score** to the corresponding assignment **for both the copy and the source**.

1. **Do you believe the solving a problem in a particular algorithmic step requires programming language skills? Support your opinion.**

   프로그래밍 언어 기술이 문제를 특정 알고리즘 단계로 해결하는 것에 꼭 필요하지는 않다. 왜냐하면 알고리즘은 특정 프로그래밍 언어가 아니라 다양한 방식으로 표현될 수 있기 때문이다. 일상 언어, 그림 뿐만 아니라 수식 또는 다이어그램(순서도), 의사 코드로도 알고리즘을 표현할 수 있다. 하지만 우리가 오늘날 보통, 일상 언어 대신 수학을 수식으로 기술하고, 알고리즘을 프로그래밍 언어로 기술하는 이유는 형식 언어(Formalized language)의 편리함에 있다. 논리학자들이 정확한 논리의 기술을 위해서 자연 언어의 모호성을 피해 유일한 독해가능성(Unique readability)를 가진 형식 언어 체계인 1차 술어 논리(First-order predicate logic) 언어를 만들어냈듯이, 알고리즘을 나타내고 이행하는데 모호성을 지닌 일상 언어를 사용할 수도 있지만, 보통 정의된 단일 동작을 실행시키는 쪽을 잘하는 기계의 측면에서 살펴봤을 때, 이러한 기계의 작동 구조를 반영하고, 원하는 알고리즘을 기술하기 위해 고안된 프로그래밍 언어는 일상 언어보다 훨씬 효율적으로 알고리즘을 기술하고 시험하는데 사용할 수 있다고 생각한다.

2. **Who is said to be the first programmer in human history? Use the Internet for help.**

   영국의 에이다 러브레이스가 인간 역사상 첫 번째 프로그래머이다. 에이다는 수학자 찰스 베비지가 고안하였던 튜링 기계인 해석 기관의 설계를 바탕으로 베르누이 수를 구하는 세계 최초의 프로그램을 작성하였다.



에이다가 작성한 베르누이 수를 구하는 해석기관용 프로그램

3. Which characteristics of programming language do you think are the most important and why?

튜링 완전성(Turing completeness)이다. 튜링 완전성은 어떤 프로그래밍 언어나 추상 기계가 튜링 기계와 동일한 계산 능력을 가진다는 의미한다. 언어란 창발적인 것이기 때문에, 그 언어가 어떤 목표를 가지고 있는지에 따라서 어떠한 가독성, 효율성이라도 가질 수 있다. 예를 들어 난해한 것을 목적으로 하는 프로그래밍 언어는 난해한 가독성도 가질 수 있고, 어떤 언어가 목표로 한 복잡성(복잡성과 성능이 상반 관계인 경우)이 필요하다면 낮은 효율성도 가질 수 있다. 다만 프로그래밍 언어에서 중요한 것은 말 그대로 프로그래밍이 가능하다는 점이고, 이 점(튜링 완전성)이 만족되는 언어라면 이론 상으로 어떠한 다른 프로그래밍 언어라도 시뮬레이션 할 수 있게 된다. 따라서 프로그래밍 언어에서 가장 중요한 것은 튜링 완전성이라고 생각한다.

또는 프로그래밍 언어론(Pearson, 10판 8P)에서 제시한 프로그래밍 언어의 특성(단순성, 직교성, 데이터 타입, 구문 설계, 추상화 지원, 표현력, 타입 검사, 예외 처리, 제한된 별칭)에서 한가지 특성을 고르자면 "단순성"이 가장 중요한 특성이라고 생각한다. 왜냐하면 단순성은 프로그래밍 언어의 판독성과 작성력에 상당한 영향을 미칠 뿐 아니라, 접근성(언어 학습자의 학습 곡선 등), 신뢰성 등의 프로그래밍 언어의 중

요한 여러 요소들까지 큰 영향을 주기 때문이다.

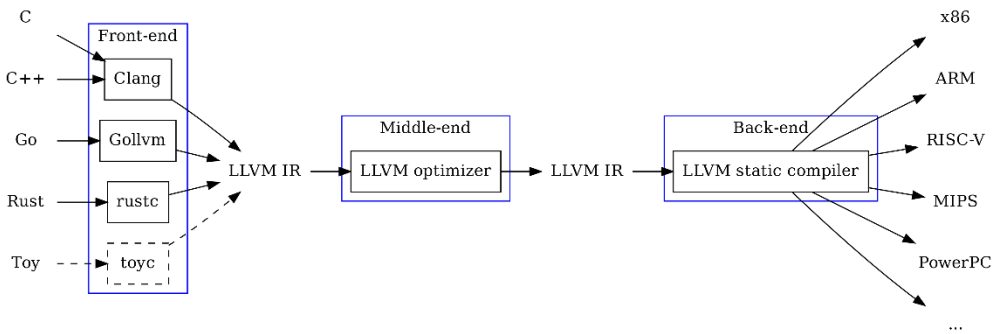
4. Explain how the orthogonality in a programming language is closely related to simplicity.

상대적으로 적은 개수의 기본 연산과 언어 구조의 조합으로 프로그램을 구축할 수 있는 언어를 직교적이라고 한다. 프로그래밍 언어가 더 직교적일수록, 프로그램을 작성하는 데 필요한 예외 규칙의 수가 줄어들고, 적은 예외 규칙의 수는 곧 단순성과 직결되기 때문에 직교성과 단순성은 밀접한 관계가 있다고 말할 수 있다.

5. Describe some design trade-offs between security and modularity in some language you know.

Java의 Reflection 기능이 그러하다. 컴퓨터 과학에서 Reflection이란 런타임 시간에 자체 구조와 동적을 검사하고, 수정할 수 있게 하는 개념이다. Python, JavaScript 등 Reflection 기능을 지원하는 언어는 많이 있지만, 특히 Java에서는 이 Reflection 기능을 이용하여 모듈성을 높여 유연한 개발을 가능케 해주는 Spring, Hibernate와 같은 Framework등이 많이 이용된다. 하지만 Reflection 개념은 보안과 상충될 수 있는 개념으로(캡슐화된 멤버 변수에 의도치 않은 접근을 허용케 하는 등) 이런 Reflection 기능을 사용한 자바의 Framework를 사용하는 경우 모듈성을 위해 보안과 어느정도 설계 절충점을 가졌다고 생각된다.

6. Can we call any programming language complete, in your opinion? Why or why not?



여러 프로그래밍 언어가 각자 목표로 하는 "완벽"에 도전하지만, 내가 생각하는 완벽에 가까운 프로그래밍 언어란 모든 프로그래밍 언어를 기반으로서 지탱해줄 수 있는 동시에, 기계어보다는 높은 추상성을 가져야 한다고 생각한다. 왜냐하면 환원주의적으로 생각하였을 때 그러한 프로그래밍 언어라면, 이보다 더 높은 프로그래밍 언어들을 모두 그 언어안에서 구현할 수(사용할 수) 있기 때문이다. 현존하는 언어 중에서 이러한 형태의 가까운 것을 말하자면 LLVM의 IR(Intermediate Representation, 중간 표현)이 그 대상이 될 수 있다고 생각한다. LLVM은 임의의 프로그래밍 언어의 정적 및 동적 컴파일을 모두 지원할 수 있는 현대적 컴파일 전략을 제공하는 목표를 가진 연구 프로젝트로 이 프로젝트에서는 여러 언어(C, C++, Rust, Go)를 LLVM IR이라는 중간 표현으로 번역하고, 이를 최적화한 뒤 각 기계에 상응하는 기계어를 생성해준다. LLVM의 IR은 내가 생각하는 조건을 실제로 어느정도 만족하기 때문에 나는 이것이 완벽에 가까운 프로그래밍 언어라고 생각한다.

7. **Was the first high-level programming language you learned implemented with a pure interpreter, a hybrid implementation system, or a compiler?**

내가 가장 처음 배운 고급 프로그래밍 언어는 C 언어(C99)로 컴파일러(GCC)로 구현되었다.

8. **Describe the advantages and disadvantages of using different computer architecture for writing a program.**

프로그램을 작성하기 위해 다른 컴퓨터 아키텍처를 사용하는 경우 장점은 프로그램을 각 아키텍처에 꼭 맞게 최적화하거나 호환성을 확보할 수 있다는 점이고, 단점은 서로 다른 컴퓨터 아키텍처간 구조의 차이로 명령 비호환이나 오류 가능성이 생긴다

는 것이다. 또한 이러한 문제를 해결하기 위해 개발에 고비용이 소요될 수 있다는 문제점도 있을 수 있다.

9. **Does a Just-In-Time (JIT) implementation system affect the execution of code? Why or why not?**

   JIT 구현 시스템은 프로그램 시작 시 중간 언어(Java Bytecode, CIL 등) 컴파일된 프로그램을 다시 기계어로 컴파일한다. 따라서 주어진 프로그램 코드를 실행할 때, 기계어 컴파일 과정 없이 실시간 번역을 통해 실행되는 인터프리터 언어에 비해 빠른 속도로 동작한다. (정적 컴파일된 언어에 비해서는 느릴 수 있으나, 기계에 종속적인 바이너리가 생성되는 정적 컴파일에 비해 컴파일(JIT 컴파일, 동적 번역)이 CPU나 운영체제에 따라 다르게 진행될 수 있는 점을 장점으로 가진다.)

10. **Some programming languages – for example, SQL- use "=" to check the equality of two expressions, while C uses it to assign values to variable. Which of these, in your opinion, is most natural and least likely to result in syntax errors? Support your answer.**

    C 스타일의 언어들이 편리함을 위해, "="를 대입 연산자로, "=='를 동등 비교 연산자로 사용하지만 나는 SQL과 같이 "="를 동등 비교 연산자로 사용하는 것이 가장 자연스럽다고 생각한다. 왜냐하면 수학적으로 "=" 기호는 동등 비교의 의미로 해석되기 때문에, 이런 관례를 모르는 프로그래밍 입문자에게도 구문 오류 가능성을 줄이고 자연스럽게 받아들여질 수 있기 때문이다. 실제로도 ALGOL의 영향을 받은 언어(Pascal, Ada)들은 ":="를, APL의 영향을 받은 언어(R)들은 "<-"를 대입 연산자로 할당하고 "="를 동등 비교 연산자로 사용한다. 이 언어들의 연산자 정의에서도 "="는 동등 비교 연산자로 사용하는 철학을 엿볼 수 있다.

<The End of the Assignment>