SNU 010.142 컴퓨터의 기초, 2006 봄
기말고사

학번: 이름:

Problem 1 [20 × (4pts, -3pts)] O/X로 답하라.

1. 디지탈 컴퓨터로 풀 수 없는 문제는 없다.
2. 디지탈 컴퓨터는 전기로만 작동된다.
3. C 프로그램은 선언(declarations)과 명령(command)으로 구성된다.
4. C 프로그램에서 이름 지을 수 있는 것은 변수와 프로시져와 타입이다.
5. C 프로그램에서는 같은 이름을 다른 용도로 쓸 수 있다.
6. C 프로그램에서 int 변수를 선언하면 8bytes의 메모리가 할당된다.
7. C 프로그램에서 int 변수가 가질 수 있는 정수 범위는 -2^{31}에서 2^{32}이다.
8. C 프로그램에서 unsigned int 변수가 가질 수 있는 정수 범위는 0에서 $2^{32}-1$이다.
9. C 프로그램에서 float 변수로 선언한 변수에는 한정된 숫자만 저장할 수 있기 때문에, C 계산식이 그 의도와는 동떨어진 매우 엉뚱한 결과를 계산할 수 있다.
10. C 에서 int, char, float는, 기본적으로 주어진 데이타 타입이다.
11. C 에서 기본적으로 주어진 타입이외의 복잡한 구조의 타입에 해당하는 데이타를 표현하기 위해서는 필요한 양의 메모리를 직접 할당받아 사용해야 한다.
12. C 에서 `malloc(sizeof(int))`는 8 bytes의 메모리를 할당하고 그 메모리의 시작 주소를 돌려준다.
13. 임의의 정수 리스트를 만들 수 있으려면 단 두개의 방법을 반복해서 이용하면 충분하다.
14. 한 방법은 빈 리스트를 만드는 방법이고, 다른 한 방법은 주어진 리스트 두개를 연결해 주는 방법이다.
15. 디지탈 컴퓨터로 표현하는 모든 데이타 구조들은 무한히 많을 수 있지만, 몇개(유한개)의 방법들을 반복 사용해서 만들 수 있는 것 뿐이다.
16. 귀납적인 사고가 복잡한 구조의 데이타를 만드는 방법들을 고안할 때 쓰인다.
17. 재귀적인 C 프로시져는 귀납적으로 구축된 데이타를 입력으로 받을 때 자연스럽게 정의된다.
18. 다음의 C 프로그램에서 `add`라는 이름이 두 가지 다른 것을 뜻하고 있고 각각의 유효범위에 따라서 제대로 분별 될 수 있다.

```
int s;
int add(int x) { return x+1;}
void main() {
   int add = 10;
   s = 0;
   add = add + add(s);
}
```

19. 다음의 C 프로그램은 실행중에 최대 68 bytes의 메모리가 필요하다.

```
typedef struct {int v;} obj;
typedef struct {obj *fst; obj *snd;} pair;
void main() {
  pair *x; obj *j; int i;
  for (i=0; i<5; i++)
    {x = malloc(sizeof(pair));
     j = malloc(sizeof(obj));
     j->v = 0; x->fst = j; x->snd = x->fst;
    }
}
```

20. 컴퓨터로 풀 수 있는 문제중에는 현재 최고 속도의 디지탈 컴퓨터가 지구상의 모든 메모리를 동원해서 풀어도 태양계 수명 보다 더 많은 시간이 필요한 것이 있다.

Problem 2 [20pts] 다음은 *The Pattern on the Stone: the simple ideas that make computers work* 의 일부분이다. 모든 빈칸에 공통으로 들어갈 한 단어는? 컴퓨터 과학/공학의 핵심 개념이다.

[pp.18–19]

Naming the two signals in computer logic 0 and 1 is an example of functional _____. It lets

us manipulate information without worrying about the details of its underlying representation. Computers are built up of a hierarchy of such functional [redacted]s, each one embodied in a building block. The blocks that perform functions are hooked together to implement more complex functions, and these collections of blocks in turn become the new building blocks for the next level.

This hierarchical structure of [redacted] is our most powerful tool in understanding complex systems, because it lets us focus on a single aspect of a problem at a time. For instance, we can talk about Boolean functions like And and Or in the abstract, without worrying about whether they are built out of electrical switches or sticks and strings or water-operated valves. For most purposes, we can forget about technology. This is wonderful, because it means that almost everything we say about computers will be true even when transistors and silicon chips become obsolete.

[pp.58–59]

We are now in a position to summarize how a computer works, from top to bottom. Most readers will have lost tract of the details, but *remember that it is not important to remember how every step works!* The important thing to remember is the hierarchy of functional [redacted]s.

The work performed by the computer is specified by a *program*, which is written in a *programming language*. This language is converted to sequences of *machine-language* instructions by *interpreters* or *compilers*, via a predefined set of subroutines called the *operating system*. The instructions, which are stored in the *memory* of the computer, define the operations to be performed on data, which are also stored in the computer's memory. A *finite-state machine* fetches and executes these instructions. The instructions as well as the data are represented by patterns of *bits*. Both the finite-state machine and the memory are built of storage *registers* and *Boolean logic blocks*, and the latter based on simple *logical functions*, such as *And*, *Or*, and *Invert*. These logical functions are implemented by *switches*, which are set up either *in series* or *in parallel*, and these switches control a physical substance, such as water or electricity, which is used to send one of two possible signals from one switch to another: *1* or *0*. This is the hierarchy of [redacted] that makes computers work.

수고하셨습니다. 즐겁고 보람찬 첫 방학을 보내기 바랍니다.