1. Consider the following Prolog program.

```
takes(jane_doe, his201).
takes(jane_doe, cs254).
takes(ajit_chandra, art302).
takes(ajit_chandra, cs254).
classmates(X, Y) :- takes(X, Z), takes(Y, Z).
```

What does the query `classmates(jane_doe, X)` return? Give details of how the search procedure produces this result.

2. Consider the following C-like program.

```
int w = 3;
int x = 10;

int incw() { return ++w; }
int incx() { return ++x; }

void foo(y, z){
    printf("%d\n", y + y);
    x = 1;
    printf("%d\n", z);
}

int main() {
    foo(incw(), incx());
    return 0;
}
```

What does it print if the language uses

(a) Applicative-order evaluation?
(b) Normal-order evaluation?

3. In an assembly-language-like notation (e.g., use MIPS or a pseudocode of your own choosing), write what a good optimizing compiler would produce for the following two `switch` statements:

```
switch (a) {
case 1: x = 3; break;
case 2: x = 5; break;
case 3: x = 15; break;
case 4: x = 20; break;
case 5: x = 23; break;
default: x = 28; break;
}
```

```
switch (b) {
case 1: x = 3; break;
case 10: x = 5; break;
case 100: x = 15; break;
case 1000: x = 20; break;
default: x = 25; break;
}
```

4. For a 32-bit little-endian processor with the usual alignment rules, show the memory layout and size in bytes of the following C types.

```
union {
    struct {
        int a; /* 32-bit */
        char b; /* 8-bit */
    } s;
    int c;
} u1;

struct {
    char a;
    short b;
    int c;
    char d;
} s1;

struct {
    char a;
    char d;
    short b;
    int c;
} s2;
```