CSC 173
Homework 2
25th September, 2006
Due in class on Thursday, September 28th, 2006

1. In class, we used an example database from the book with the relations CSG, SNAP, CDH, and CR. The book also defines a 5th relation, namely CP, for course/prerequisite (see p.408). Using these relations, write a relational algebra query that answers the questions: what grade(s) did C.B. receive in the prerequisite for CS101.

2. Write a query that yields the same result as your answer to the first question, but in which you push selection and projection as far down as they can go. Which, if any, of these pushed operations are not meaningful and why.

3. Consider the following query on the bank database that was part of Quiz 1, i.e., the names of all customers who have a loan, where the loan amount is greater than any account balance they might have. What is the optimized query? Explain why (or why not) the selects and projects can be pushed down into the individual relations.

4. The following code will crash/dump core if you were to execute it on a real machine. Why? How would you fix it (using either static variables or dynamic memory allocation)?

```
int *a = 0;
*a = 5;
```

5. If you declare a local variable that's a C array, you have to specify the number of elements in the array. On the other hand, if you declare a function parameter that's a C array, you don't have to specify the number of elements, e.g.

```
void foo (int parameter_array[])
{
    int local_array[10];
    ...
```

   Why the difference?

6. What does this program print in C?

```
#include "stdio.h"

struct integer {
    int val;
};

static void bar(struct integer i) {
    i.val = 5;
}
```

```
main(int argc, char *argv[]) {
    struct integer n;
    n.val = 1;
    bar(n);
    printf("%d\n", n.val);
}
```

What does this similar program print in Java?

```
import java.io.*;

class integer {
    public int val = 0;
}

public class num {

static void bar(integer i) {
    i.val = 5;
}

public static void main(String[] args) {
    integer n = new integer();
    n.val = 1;
    bar(n);
    System.out.println(n.val);
}
}
```

Why the difference? How could you modify the C program to make it behave like the Java one?