

Examination 2 Programming Languages and Systems Concepts,
CSI 511
Fall 2001 (December 17, 2001)

1 Rules of the Exam

This examination is open book and notes. Calculators are permitted. Networked devices are strictly prohibited. The questions are marked as to their relative value, the exam will be scored out of 100% but is worth 25 points toward your course grade. Relax and try to do what you can.

2 The Problem Set

1. Readings (10 %) Note: Word for word copying from the readings will not get credit, answer IN YOUR OWN WORDS:
 - (a) What is step-wise refinement and what problem does Niklaus Wirth apply it to?(5 %)
 - (b) Dijkstra claims go-to statements are bad for higher level languages, because they make programs hard to understand and are not more expressive than go-to free constructs. Jacopini had an automated translation system to take programs containing go-to's and create go-to free programs. Does this solve the problem, why or why not?(5 %)?
2. Array Management (25 %):
 - (a) Consider a three dimensional array declared as:

```
A : array [1..3,1..4,1..5] of integer;
```

using a Pascal like syntax. Assuming that integers are 32 bit values, and that the array starts at memory location 1000 (base 10) what is the offset of $A[1, 2, 3]$ if A is stored in row major order? What is the offset if column major order is used (5 %), show your work?
 - (b) The Perl programming language has associative arrays, in C++ associative arrays are not part of the language, but they are supported as part of the STL library. STL supports associative arrays via the “map” container class. STL maps are implemented using binary search trees (red-black trees to be exact, a balanced variety of binary search tree). Consider an STL map containing N elements.
 - i. Give the computational complexity of finding an element in a C++ map. Justify your answer (5 %).
 - ii. Give the cost of traversing a C++ map. Justify your answer (5 %).

- (c) Fortran 90 supports array slicing, some high performance computing languages support slices in a more general sense. Suppose that for a matrix, we wanted to make a *diagonal* array slice across an $N \times N$ array of 64 bit (double precision) floating point values. If the array begins at location 2000 and the default Fortran ordering is used, give the calculation the compiler would use to access the i th element, where $1 \leq i \leq N$ (10 %).
3. Object oriented computing (20 %): Consider the following C++ code using templates, which exploits polymorphism and operator overloading. The code is supposed to transpose a dynamically allocated two dimensional array, and use the transpose of the array. However, when running the program, the performance is extremely poor, and it gives the wrong results. Find and fix the bugs in this program.

```

template <class T> class Array2D {
private:
    unsigned int LENGTH; // Number of elements in array
    unsigned int nrows;
    unsigned int ncols;
    unsigned int rowsize;
    T *array;
public:
    Array2D( const int row, const int col ){
        this->LENGTH = row * col;
        this->nrows = row;
        this->ncols = col;
        this->array = new T[this->LENGTH];
    }
    Array2D( const Array2D<T> &x ){
        if (this != &x){
            if (this->T != NULL){
                delete [] this->array;
            }
            this->array = new T[x.LENGTH];
            this->nrows = x.nrows;
            this->cols = x.ncols;
            for (int i = 0; i < this->Length; ++i){
                this->array[i] = x.array[i];
            }
        }
    }
    unsigned int len() { return LENGTH; }
    unsigned int rows() { return nrows; }
    unsigned int cols() { return ncols; }
    // 2D Indexing Function into 1D Data Storage Structure
    inline T& operator() ( const register int y, const register int x ) const {
        int pos = // The students fill this in
        return array[pos];
    }
}

void transpose_swap(Array2D<int> data, int i , int j){
    int tmp = data(j,i);
    data(j,i) = data(i,j);
    data(i,j) = tmp;
}

```

```

void transpose( Array2D<int> data ){
    for (int i = 0; i < data.rows(); ++i){
        for (int j = 0; j < i; ++j){
            transpose_swap(data, i, j);
        }
    }
}

extern void use_transpose(Array2d<int> A);

int main(){
    Array2D<int> A(1000,1000);

    for (int i = 0; i < data.rows(); ++i){
        for (int j = 0; j < data.cols(); ++j){
            cin >> A(i,j); // read in A(i,j) from the input
        }
    }
    transpose(A);
    use_transpose(A);
}

```

4. Parameter transmission and Scope rules (30 %):

(a) Consider the following program:

```

int a = 1;
int b = 2;
int c = 3;

program main()
    procedure p(int x, int y)
        procedure q(int x, int y)
            begin
                x = c + 1;
                print(a, b, c, x, y);
            end
        begin
            int c = 4;
            x = a + 5;
            y = b + 6;
            q(a, b);
            print(a, b, c, x, y);
        end
    begin
        int x = 8;
        int y = 9;
        p(x,y);
        printf(a, b, c, x, y);
    end

```

- i. Assume Static Scope: Give the output of this program for: Call by Value, Call by Reference, Call by Value-Result and Call by Name (10 %).
- ii. Assume Dynamic Scope: Give the output of this program for: Call by Value, Call by Reference, Call by Value-Result and Call by Name (10 %).

(b) Consider the following program (assume C style scoping):

```
int x = 3;
int y = 4;
int m = 5;

int
mult_mod(int a, int b, int c){
    return (a * b) mod c;
}

int
main(){
    int product = multmod(x + 1, y + 2, m);
    print(product, x, y, m);
}
```

What is the output of this program using call by value, call by value result, call by reference and call by name (10 %)?

5. Garbage Collection (20 %):

- (a) Consider a doubly linked list, which is circularly linked. The program managing this list has a single pointer to the head of the list. You may assume that there are no other pointer to any nodes in the list. Suppose that the user sets the pointer to NULL, to reset the list to be empty.
 - i. Would mark and sweep approaches be able to reclaim this space, why or why not (5 %)?
 - ii. Would reference counting approaches be able to reclaim this space, why or why not (5 %)?
- (b) Your company hires a new programmer, Wile E. Coyote (Super Genius) who suggests that the next and previous pointers can be consolidated by using a trick he found on the web which works by using exclusive or to encode the pointers. The trick involves keeping the address of the head and the tail of the list directly recorded in pointers. The pointer manipulations seem to allow for correct traversal of the list, but he wants to know whether a mark and sweep or reference counting approach can handle the garbage collection properly. What do you think the answer is, and why (10 %)?