

Programming Languages: Assignment 1

G22.2110

Due in 2 weeks: February 22 2000

To be sent by email to crutcher@cs.nyu.edu and mauny@cs.nyu.edu.

Attachments are welcome.

Note: You are required to include a printout of a session showing a run of your program(s) on some **general** test data.

Late homeworks will not be accepted (except in the event of unavoidable circumstances). If for some reason you will be unable to hand in a homework on time, please discuss it with me beforehand.

Problem 1 From “Sethi: Programming Languages: Concepts and Constructs, 2nd Ed.”,

- exercise 2.12 (p.51, referring to fig. 2.9, p.45);
- exercise 2.19 (p. 52).

Problem 2 Each of the following program fragments is the body of a function “int f(int x)”. In each case, give the values of f(-1), f(0), and f(1).

1. if (x < 0) return -1;
 else if (x = 0) return 0;
 else return 1;
2. if (x > 0)
 return 1;
 return 0;
3. if (x != 0);
 return -1;
 return 0;
4. for (; x > 0; x++) {
 if (x == -1) continue;
 if (x == 0) break;
 }
 return x;

Problem 3 Write a C function:

```
int substr(char *pat, char *str)
```

that returns the index of the first occurrence of the string pat in str, and -1 if pat does not occur in str. (A naive algorithm is fine.)

Problem 4 *In some situations, it is preferable to represent ordered sets of integers as integers stored in a binary search tree (compared to an array or a list). For instance, when integers have to be added and deleted from the set, a tree is more flexible than an array, and more efficient than a list.*

A binary search tree (bs-tree, for short) is a binary tree such that at any node n , the value $n.v$ is

- *greater than any value from the nodes in the left subtree, and*
- *smaller than any value from the nodes in the right subtree.*

Binary search trees give access to a particular element in $O(\log(N))$, in the best case, where N is total the number of elements.

Write in C the following parts of an binary search tree program:

1. *type definitions;*
2. *a function adding an integer to a bs-tree;*
3. *a function deleting an integer from a bs-tree (the resulting tree must be a bs-tree);*
4. *a function listing in increasing order all integers from a bs-tree.*