Types, Part I

Programming Languages
CS 214

Types

A *type* is _____.

Examples from C++:

- The _____ type:
  V = {INT_MIN, ... -1, 0, 1, ..., INT_MAX-1, INT_MAX}
  O = {<<, >>, +, -, *, /, %, =, ++, --, ...}
- The _____ type:
  V = {NUL, ..., '0', …, '9', …, 'A', ..., 'Z', ..., 'a', ..., 'z', DEL}
  O = {<<, >>, =, ++, --, isupper(), islower(), toupper(), … }
- The _____ type:
  V = {"", "A", "B", "C", ..., "AA", "AB", "AC", ..., "AAA", ...}
  O = {<<, >>, +, +=, [], find(), substr(), … }

Fundamental Types

Let's assume the existence of some basic types:

Name	V	C++	Ada	Smalltalk	Lisp
_____	false, true	*bool*	*boolean*	*Boolean*	*boole*
_____	the set of chars	*char*	*character*	*Character*	*character*
_____	the integers	*int*	*integer*	*Integer*	*integer*
_____	the reals	*double*	*float*	*Float*	*real*

Given these, new types can be created via _____.

Each constructor has 3 components:

- The syntax used to denote that constructor;
- The set of elements produced by that constructor; and
- The operations associated with that constructor.

Set Constructor I: _____

The product constructor is the basis for _____.

- The product of two sets A and B is denoted _____.
- A×B consists of all ordered pairs (a, b): $a \in A, b \in B$.

  A×B×C consists of all ordered triples (a, b, c): $a \in A, b \in B, c \in C$.

  A×B× … ×N consists of all ordered n-tuples (a, b, …, n): $a \in A, b \in B, \ldots, n \in N$.

  Example: the set _____ has 256 elements:
  { …, (true, 'A'), (false, 'A'), (true, 'B'), (false, 'B'), …, }.
- Operations associated with product are the _____ operations:
  - *first*, applied to an n-tuple $(s_1, s_2, \ldots, s_n)$ returns ____.
  - *second*, applied to an n-tuple $(s_1, s_2, \ldots, s_n)$ returns _____.
  - *nth*, applied to an n-tuple $(s_1, s_2, \ldots, s_n)$ returns _____.

Product Example: C++ structs

```
struct Student
{
   int id;
   double gpa;
   char gender;
};
Student aStudent;
```

Formally, a Student consists of:

_____

Formally, a particular Student:

```
aStudent.id = 12345;
aStudent.gpa = 3.75;
aStudent.gender = 'F';
```

is the 3-tuple: _____.

The C++ _____ is a projection operation:

```
cout << aStudent.id          // extract id
     << aStudent.gpa         // extract gpa
     << aStudent.gender      // extract gender
     << endl;
```


Set Constructor II: _____

The function constructor is the basis for _____.

– The set of all functions from a set A to a set B is denoted _____.
– A particular function *f* mapping A to B is denoted _____.

Examples:

– The set (char) → bool contains all functions that map char values into bool values, some C examples of which include:

_____ _____
_____ _____
_____ _____

– The set (char) → char contains all functions that map char values into char values, some C examples of which include:

_____ _____

Function and Product

What does this set contain? $(\text{int} \times \text{int}) \rightarrow \text{int}$

– _____.

Examples? _____ _____
_____ _____

Suppose we define an aggregate named *IntPair*:

```
struct IntPair {
   int a,
       b;
};
```

and then define a function named add():

```
int add(IntPair ip) {
  return ip.a + ip.b;
};
```

add() is a member of the set: _____

– The function constructor let us create _____.

Function Arity

Product serves to denote an aggregate or an argument-list.

What does this set contain? $(\text{int} \times \text{int}) \rightarrow \text{bool}$

– All functions that map _____.

Examples? _____ _____
_____ _____

Definition:

The number of operands an operation requires is its _____.

– Operations with 1 operand are _____ operations, with *arity-1*.

– Operations with 2 operands are _____ operations, with *arity-2*.

– Operations with 3 operand are _____ operations, with *arity-3*.

– ...

Example Ternary Operation

The C/C++ conditional expression has the form:

<expr>$_0$? <expr>$_1$: <expr>$_2$

producing <expr>$_1$ if <expr>$_0$ is true, and
producing <expr>$_2$ if <expr>$_0$ is false.

Here is a simple *minimum()* function using it:

```
int minimum(int first, int second) {
  return (first < second) ? first : second;
};
```

The C/C++ conditional expression is a ternary operation, which in this case is a member of the set:

_____

Operator Positioning

Operators are also categorized by their position relative to their operands:

- _____ operators appear *between* their operands: `1 + 2`
- _____ operators appear *before* their operands: `+ 1 2`
- _____ operators appear *after* their operands: `1 2 +`

_____ ≡ _____ ≡ _____

Prefix, infix, and postfix notation are different conventions for the same thing; a language may choose any of them:

C++ Expr	Category	Value	Lisp Expr	Category	Value
x < y	_____	true, false	(< x y)	_____	true, false
++x	_____	x+1	(incf x)	_____	x+1
11 + 12	_____	23	(+ 11 12)	_____	23
!*flag*	_____	neg. of *flag*	(not *flag*)	_____	neg. of *flag*
cout << x	_____	cout	(princ x *str*)	_____	x
x++	_____	x	None		

Set Constructor III: _____

Kleene Closure is the basis for representing _____.

– The Kleene Closure of a set A is denoted _____.

– The Kleene Closure of a set is the set of all tuples that can be formed using elements of that set.

Example: The Kleene Closure of bool -- bool* -- is the infinite set:

{ _____

_____ … }

– For a tuple t ∈ A*, the operations include:

null(A*) → bool

```
null(())          → _____
null((false))     → _____
null((true))      → _____
```

first(A*) → A

```
first((true, false))   → _____
first((false, true))   → _____
```

rest(A*) → A*

```
rest((true, true, false)) → _____
rest((false, true, true)) → _____
```


Kleene Closure Examples

If *char* is the set of ASCII characters, what is *char** ?

– The infinite set of _____.

(AKA the set of all _____).

The C/C++ notation: "Hello"

is just a different syntax for: ('H', 'e', 'l', 'l', 'o')

Thus, *int** denotes a sequence (array, list, …) of _____;

```
int intStaticArray[32];
int * intDynamicArray = new int[n],
vector<int> intVec;
list<int> intList;
```

*real** denotes a sequence (array, list, …) of _____;

and so on.

Sequence Operations

Sequence operations can be built via *null(), first()*, and *rest()*

- An output operation can be defined like this (pseudocode):

```
void print(ostream out, int * a) {
  if ( !null(a) ) {
    out << first(a) << ' ';
    print(out, rest(a));
  }
};
```

- A subscript operation can be defined like this (pseudocode):

```
char & operator[](int * a, int i) {
  if (i > 0)
    return operator[](rest(a), i-1);
  else
    return first(a);
};
```

In Lisp:
first is called ____
rest is called ____.

Practice Using Constructors

Give formal descriptions for:

- The *logical and* operation (&&):
  - How many operands does it take? ___
  - What types are its operands? _____
  - What type of value does it produce? _____
  - So && is a member of _____
- The C++/*STL substring* operation (*str*.substr(*i*,*n*)):
  - How many operands does it take? ___
  - What types are its operands? _____
  - What type of value does it produce? _____
  - So substr() is a member of: _____
- For you: The *logical negation* operation (!):

More Practice

- For you: this *C++ record:*

```
struct Student {
  int myID;
  string myName;
  bool iAmFullTime;
  double myGPA;
};
```

- For you: an *accessor* method:

```
struct Student {
  int myID;
  int id() const ;
  string myName;
  bool iAmFullTime;
  double myGPA;
};
```

- How does this affect our *Student* description?

More Practice (ii)

- For you: A "complete" class:

```
class Student {
 public:
  Student();
  Student(int, string, bool, double);
  int getId() const;
  string getName() const;
  bool getFullTime() const;
  double getGPA() const;
  void read(istream &);
  void print(ostream &) const;
 private:
  int    myID;
  string myName;
  bool   iAmFullTime;
  double myGPA;
};
```


Summary

A type consists of _____ and _____.

The set constructors:

- _____,
- _____, and
- _____

provide a formal way to represent types:

→ Use the *product* and *Kleene closure* to represent the ____;

→ Use the *function* constructor to represent the _____ on the type.