

1. [10] Give the type of the following Haskell expression

`\x y z -> case x of [] -> y<z; [0] -> y>z`

Answer:

`(Num a, Ord b) => [a] -> b -> b -> Bool`

2. [10] Suppose that sets of letters (both uppercase and lowercase) are to be implemented in C by arrays of `int`. Assuming that a `char` consists of 8 bits and that `size(int) = 4`, what is the minimum size of these arrays?

The cardinality of the universe is 52. Each element corresponds to one bit in the array, and each array element contains $4 \times 8 = 32$ bits, so the smallest array size is **2** (for 64 bits).

3. [10] For each of the following C expressions, indicate whether it is true for *all* values of *a* and *b*, assuming left-to-right evaluation of operands, by *circling* (L→R).

If it is true for all values of *a* and *b* assuming right-to-left evaluation of operands, circle (L←R).

If it is true for all values of *a* and *b* regardless of the order of evaluation of operands, circle both (L→R) and (L←R).

- | | | |
|-------------------------------------|---|---|
| a. <code>(a += b) == (a + b)</code> | L→R | <input checked="" type="checkbox"/> L←R |
| b. <code>(a = b) == (b = a)</code> | <input checked="" type="checkbox"/> L→R | <input checked="" type="checkbox"/> L←R |
| c. <code>++a == a++</code> | <input checked="" type="checkbox"/> L→R | L←R |
| d. <code>a > a++</code> | L→R | <input checked="" type="checkbox"/> L←R |
| e. <code>a++ == a</code> | L→R | <input checked="" type="checkbox"/> L←R |

4. [10] Rewrite the following C function

```
int rsum( int A[], int n, int s )
{
    int sum = 0;
    for( int k = 0; k < n; k++ )
    {
        sum += A[k];
        if( sum >= s ) return k;
    }
    return (-1);
}
```

using pointers instead of array indexing. Assume that `int` and pointer values each occupy 32 bits.

Do not change the function's type. For full credit, you must not use an integer counter in the loop.

```
int rsum( int A[], int n, int s )
{
    int sum = 0;
    int* pLim = A + n;
    for( int* p = A; p < pLim; p++ )
    {
        sum += *p;
        if( sum >= s ) return (p - A);
    }
    return (-1);
}
```

5. [8] Predict the output of the following program

```

function alpha( integer n ) : integer
{
    integer t = n+2; n := n+10; return t;
}
function beta( integer r, s ) : integer
{
    return alpha(r) + s + s;
}
procedure main()
{
    integer a = 5;
    integer b = beta( a, alpha(a) );
    print a;
}

```

assuming that arguments are bound to parameters

- a. by value 5

By-value binding means that updating a parameter has no effect on the argument to which it is bound.

- b. by name 35

beta(a, alpha(a))	where a = 5
↪ alpha(a) + alpha(a) + alpha(a)	where a = 5
↪ 7 + alpha(a) + alpha(a)	where a = 15
↪ 7 + 17 + alpha(a)	where a = 25
↪ 7 + 17 + 27	where a = 35
↪ 51	where a = 35

6. [12] Predict the output of the following program

```

integer x = 10, y = 20, z = 30;
procedure f( integer a, integer b )
{
    a := 1; b := 2; z := 5;
}
procedure main()
{
    f( x, z ); print x, y, z;
}

```

assuming that arguments are bound to parameters

- a. by value 10 20 5

- b. by reference 1 20 5

- c. by value-result 1 20 2

7. [10] Consider a Unix pipe

p1 | p2 | p3

which takes a file of phone numbers with area codes and produces a file of Austin-area numbers with their area codes stripped off. In this pipe,

- p1 produces a file of telephone numbers, one number per line; the numbers' format is:
aaa-eee-nnnn
- p2 filters out those lines whose numbers do not begin with 512
- p3 removes the 512 prefixes from the numbers

Write the function p2 in C or C++.

```
void main()
{
    char num[20];
    while( cin )
    {
        cin.getLine( num, 20 );
        if( num[0] == '5' && num[1] == '1' && num[2] == '2' )
            cout << num << endl;
    }
}
```

8. [10] Given two tasks A and B, both of which output to the standard output stream, add synchronization code to either or both tasks which ensures that the combined output of the two tasks spells "Texas".

```
task A is
begin
    put( "T" );
    put( "x" );
    put( "s" );
end A;
```

```
task B is
begin
    put( "e" );
    put( "a" );
end B;
```

A solution:

```
task A is
begin
    put( "T" );
    B.e;
    put( "x" );
    B.a;
    put( "s" );
end A;
```

```
task B is
begin
    accept e do put( "e" ); end e;
    accept a do put( "a" ); end a;
end B;
```

Another solution:

```
task A is
begin
    put( "T" );
    B.one;
    accept two;
    put( "x" );
    B.three;
    accept four;
    put( "s" );
end A;
```

```
task B is
begin
    accept one;
    put( "e" );
    A.two;
    accept three;
    put( "a" );
    A.four;
end B;
```