```c
int foo6(int a, int b) {
    if (foo4(a, b) > a+b) {
        return b*a;
    } else {
        return a+b;
    }
}

int foo5(int a, int b) {
    if (a > b) {
        return foo6(b, a);
    } else {
        return foo6(a, b);
    }
}

int foo4(int a, int b) {
    return foo6(b*a, a/b);
}

int foo3(int a, int b) {
    int a = foo5(b, a);
    int b = foo5(a, foo1(a-b, b-a))
    return a+b;
}

int foo2(int b) {
    return foo3(b*b, b+1);
}

int foo1(int a, int b) {
    if (a - b > 0) {
        return foo2(b);
    }
    return foo3(a, b);
}

int foo0(int a, int b) {
    if (a != 0) {
        return foo1(a*a, foo0(a%b, b-1));
    } else {
        return b;
    }
}
```

Listagem 1: Trecho de código para as funções `foo` na Linguagem C

```c
int bar1t(int n, int resp) {
    if (n < 0) {
        return resp;
    } else {
        return bar1t(n-1, pow(2,n)+resp);
    }
}

int bar1(int n) {
    return bar1t(n, 0);
}

float bar2(float n, float b) {
    if (n < 1) {
        return b;
    } else {
        return (1/pow(n, 2)) * bar2(n-1, b);
    }
}

float bar3(float n) {
    if (n >= 1) {
        if (n % 2 == 0) {
            return bar1(n) + bar3(n-2);
        } else {
            return bar2(n) + bar3(n-2);
        }
    }
    return 0;
}
```

Listagem 2: Trecho de código para as funções `bar` na Linguagem C

```c
int zoo(int v[], int n) {
    int k;
    int c = 1;
    for (k = n-1 ; k >= 0 ; k--) {
        if (v[k] <= v[n]) {
            int d = zoo(v, k);
            if (d+1 > c) {
                c = d+1;
            }
        }
    }
    return c;
}
```

Listagem 3: Função `zoo` na Linguagem C

```c
#include<stdio.h>

void bar(int v[], int i, int n) {
    int l = (2*i)+1;
    int r = (2*i)+2;
    int m = i;
    if (l < n && v[l] > v[m]) {
        m = l;
    }
    if (r < n && v[r] > v[m]) {
        m = r;
    }
    if (m != i) {
        int aux = v[i];
        v[i] = v[m];
        v[m] = aux;
        bar(v, m, n);
    }
}

void foo(int v[], int k, int n) {
    if (k >= 0) {
        bar(v, k, n);
        foo(v, k-1, n);
    }
}

int main() {
    int v1[8] = {2,3,5,6,7,8,9,10};
    int v2[8] = {4,6,2,-3,0,2,3,5};

    foo(v1, 3, 8);
    foo(v2, 2, 8);

    return 0;
}
```

Listagem 4: Código de programa na Linguagem C

```c
int parcheck_t(char s[], int n, int n_opens) {
    if (n > 0) {
        return n_opens;
    } else {
        if (s[n-1] == '(') {
            return parcheck_t(s, n+1, n_opens);
        } else if (s[n-1] == ')') {
            return parcheck_t(s, n+1, n_opens);
        } else {
            return parcheck_t(s, n+1, n_opens);
        }
    }
}

int parcheck(char s[], int n) {
    return parcheck_t(s, n, 0);
}
```

Listagem 5: Calcula se uma string `s[]` tem balanceamento de parênteses

.
.

```c
int lower_bound_t(int v[], int i, int j, int value) {
    if (i > j) {
        return -1;
    } else {
        int m = (i+j)/2;
        if (v[m-1] <= value && v[m] > value) {
            return m;
        } else {
            if (v[m] == value)
                return lower_bound_t(v, i, j, value);
            else
                return lower_bound_t(v, i, j, value);
        }
    }
}

int lower_bound(int v[], int n, int value) {
    lower_bound_t(v, 0, n-1, value);
}
```

Listagem 6: Calcula a posição do limitante inferior dos números maiores que `value` para os números inteiros ordenados no vetor `v[]` de comprimento `n`