Queens.tig

```
make: Nothing to be done for 'all'.
[himani]$111801016-compilers make test
./tc test/queens.tig --pp
let
    var N := 8
    type intArray = array of int
    var row := intArray [N] of 0
    var col := intArray [N] of 0
    var diag1 := intArray [N + N - 1] of 0
    var diag2 := intArray [N + N - 1] of 0
    function printboard() =
        (
        for i := 0 to N - 1
            do
                (
                for j := 0 to N - 1
                    do
                        print(if col[i] = j then
                                O
                        else
                                .);
                print(\n)
                );
        print(\n)
        )

    function try(c:int) =
        (
        if c = N then
            printboard()
        else
            for r := 0 to N - 1
                do
                    if row[r] = 0 & diag1[r + c] = 0 & diag2[r + 7 - c] = 0 then
                        (
                        row[r] := 1;
                        diag1[r + c] := 1;
                        diag2[r + 7 - c] := 1;
                        col[c] := r;
                        try(c + 1);
                        row[r] := 0;
                        diag1[r + c] := 0;
                        diag2[r + 7 - c] := 0
                        )
        )
in
    (
    try(0)
    )
end
```

Merge.tig

```
./tc test/merge.tig --pp
let
    type any = {any:int}
    var buffer := getchar()
    function readint(any:any):int =
        let
            var i := 0
            function isdigit(s:string):int =
                ord(buffer) >= ord(0) & ord(buffer) <= ord(9)

            function skipto() =
                while buffer =   | buffer = \n do
                    buffer := getchar()
        in
            (
            skipto();
            any.any := isdigit(buffer);
            while isdigit(buffer) do
                (
                i := i * 10 + ord(buffer) - ord(0);
                buffer := getchar()
                );
            i
            )
        end

    type list = {first:int, rest:list}
    function readlist():list =
        let
            var any := any {any = 0}
            var i := readint(any)
        in
            (
            if any.any then
                list {rest = readlist(), first = i}
            else
                nil
            )
        end


    function merge(a:list, b:list):list =
        if a = nil then
            b
        else
            if b = nil then
                a
            else
                if a.first < b.first then
                    list {rest = merge(a.rest, b), first = a.first}
                else
                    list {rest = merge(a, b.rest), first = b.first}

    function printint(i:int) =
        let
            function f(i:int) =
                if i > 0 then
                    (
                    f(i / 10);
```

```
    function printint(i:int) =
        let
            function f(i:int) =
                if i > 0 then
                    (
                    f(i / 10);
                    print(chr(i - i / 10 * 10 + ord(0)))
                    )
        in
            (
            if i < 0 then
                (
                print(-);
                f(-i)
                )
            else
                if i > 0 then
                    f(i)
                else
                    print(0)
            )
        end


    function printlist(l:list) =
        if l = nil then
            print(\n)
        else
            (
            printint(l.first);
            print( );
            printlist(l.rest)
            )
    var list1 := readlist()
    var list2 := (
    buffer := getchar();
    readlist()
    )
in
    (
    printlist(merge(list1, list2))
    )
end
```

Test 1:

```
./tc test/test1.tig --pp
let
    type arrtype = array of int
    var arr1:arrtype := arrtype [10] of 0
in
    (
    arr1
    )
end
```

Test 2-6:

```
./tc test/test2.tig --pp
let
    type myint = int
    type arrtype = array of myint
    var arr1:arrtype := arrtype [10] of 0
in
    (
    arr1
    )
end
./tc test/test3.tig --pp
let
    type rectype = {name:string, age:int}
    var rec1:rectype := rectype {age = 1000, name = Nobody}
in
    (
    rec1.name := Somebody;
    rec1
    )
end
./tc test/test4.tig --pp
let
    function nfactor(n:int):int =
        if n = 0 then
            1
        else
            n * nfactor(n - 1)
in
    (
    nfactor(10)
    )
end
./tc test/test5.tig --pp
let
    type intlist = {hd:int, tl:intlist}
    type tree = {key:int, children:treelist}
    type treelist = {hd:tree, tl:treelist}
    var lis:intlist := intlist {tl = nil, hd = 0}
in
    (
    lis
    )
end
./tc test/test6.tig --pp
let
    function do_nothing1(a:int, b:string) =
        do_nothing2(a + 1)

    function do_nothing2(d:int) =
        do_nothing1(d, str)
in
    (
    do_nothing1(0, str2)
    )
end
```

Test 7,8,11,12:

```
./tc test/test7.tig --pp
let
    function do_nothing1(a:int, b:string):int =
        (
        do_nothing2(a + 1);
        0
        )

    function do_nothing2(d:int):string =
        (
        do_nothing1(d, str);

        )
in
    (
    do_nothing1(0, str2)
    )
end
./tc test/test8.tig --pp
if (
10 > 20
) then
    30
else
    40./tc test/test11.tig --pp
for i := 10 to
    do
        i := i - 1./tc test/test12.tig --pp
let
    var a := 0
in
    (
    for i := 0 to 100
        do
            (
            a := a + 1;
            (
            nil
            )
            )
    )
end
```

Test 14,15,42:

```
./tc test/test14.tig --pp
let
    type arrtype = array of int
    type rectype = {name:string, id:int}
    var rec := rectype {id = 0, name = aname}
    var arr := arrtype [3] of 0
in
    (
    if rec <> arr then
        3
    else
        4
    )
end
./tc test/test15.tig --pp
if 20 then
    3./tc test/test42.tig --pp
let
    type arrtype1 = array of int
    type rectype1 = {name:string, address:string, id:int, age:int}
    type arrtype2 = array of rectype1
    type rectype2 = {name:string, dates:arrtype1}
    type arrtype3 = array of string
    var arr1 := arrtype1 [10] of 0
    var arr2 := arrtype2 [5] of rectype1 {age = 0, id = 0, address = somewhere, name = aname}
    var arr3:arrtype3 := arrtype3 [100] of
    var rec1 := rectype1 {age = 44, id = 2432, address = Kapou, name = Kapoios}
    var rec2 := rectype2 {dates = arrtype1 [3] of 1900, name = Allos}
in
    (
    arr1[0] := 1;
    arr1[9] := 3;
    arr2[3].name := kati;
    arr2[1].age := 23;
    arr3[34] := sfd;
    rec1.name := sdf;
    rec2.dates[0] := 2323;
    rec2.dates[2] := 2323
    )
end
```

Test 46, 47, 48:

```
./tc test/test46.tig --pp
let
    type rectype = {name:string, id:int}
    var b:rectype := nil
in
    (
    b = nil;
    b <> nil
    )
end
./tc test/test47.tig --pp
let
    type a = int
    var b := 4
    type a = string
in
    (
    0
    )
end
./tc test/test48.tig --pp
let
    function g(a:int):int =
        a
    type t = int
    function g(a:int):int =
        a
in
    (
    0
    )
end
```