Programming Systems- Assigmnment 2

SILPA C

Ph.D.,IIIT Chittoor

February 10, 2017

Program to implement Towers of Hanoi Problem in JULIA

```
function toh(ndisks::Int64, startPeg, auxPeg, endPeg)
    if ndisks==1
        println("Move disk from peg $startPeg  to peg $endPeg")
    else
        toh(ndisks-1, startPeg, endPeg, auxPeg)
        println("Move disk from peg $startPeg  to peg $endPeg")
        toh(ndisks-1, auxPeg, startPeg, endPeg)
    end
end
toh(3,'a','b','c')
```

OUTPUT:

```
julia> include("toh1.jl")
Move disk from peg a to peg c
Move disk from peg a to peg b
Move disk from peg c to peg b
Move disk from peg a to peg c
Move disk from peg b to peg a
Move disk from peg b to peg c
Move disk from peg a to peg c
```

Program to implement Insertion sort algorithm in JULIA

```
function isort(a,n)
        for i=1:n
                temp=a[i]
                j=i
                while j>1 && a[j-1]>temp
                        a[j]=a[j-1]
                        j-=1
                end
                a[j]=temp
        end
        println("$a")

end
```

OUTPUT:

```
julia> include("insert.jl")
isort (generic function with 1 method)

julia> isort([5,3,7,2,9,1],6)
[1, 2, 3, 5, 7, 9]
```

Another way to implement INSERTION SORT in JULIA

```
julia> sort([9,12,6,13,25,4,15,7,1,3,19],alg=InsertionSort)
11-element ArrayInt64,1:
1
3
4
6
7
9
12
13
15
19
25
```

Program to implement queue operations in JULIA

```
type Que{T}
    a::Array{T,1}
end

Que() = Que(Any[])
Que(a::DataType) = Que(a[])
Que(a) = Que(typeof(a)[])

Base.isempty(q::Que) = isempty(q.a)

function Base.pop!{T}(q::Que{T})
    !isempty(q) || error("queue must be non-empty")
    pop!(q.a)
end

function Base.push!{T}(q::Que{T}, x::T)
    unshift!(q.a, x)
    return q
end

function Base.push!{T}(q::Que{Any}, x::T)
    unshift!(q.a, x)
    return q
end
```

OUTPUT: julia> include("que.jl")

julia> q=Que()
QueAny(Any[])

julia> push!(q,1)
QueAny(Any[1])

julia> push!(q,2.0)
QueAny(Any[2.0,1])

julia> push!(q,"three")
QueAny(Any["three",2.0,1])

julia> isempty(q)

```
false

julia> pop!(q)
1

julia> isempty(q)
false

julia> pop!(q)
2.0

julia> pop!(q)
"three"

julia> isempty(q)
true

julia> pop!(q)
ERROR: queue must be non-empty
in pop!(::QueAny) at /home/silpa/Desktop/que.jl:15
```

Program to implement Heap sort algorithm in JULIA

```
function heapsort(a,n::Int64)
        i=(n/2)-1
        while i>=1
                heapify(a,n,i)
                i=i-1
        end
        i=n-1
        while i>=1
                temp1=a[1]
                a[1]=a[i]
                a[i]=temp1
                heapify(a,i,1)
                i=i-1
        end
        println("$a")
end
function heapify1(a,n,i::Int64)
        hindex=i
        l=2*i+1
        r=2*i+2
        if l<n && a[l]>a[hindex]
                hindex=r
        end
        if r<n && a[r]>a[hindex]
                hindex = r
        end
        if hindex != i
                temp=a[i]
                a[i]=a[hindex]
                a[hindex]=temp
                heapify1(a, n, hindex)
        end
end
```

OUTPUT:

julia> include("hsort1.jl")
heapsort (generic function with 1 method)

julia> include("hsort.jl")
heapify (generic function with 1 method)

julia>heapsort([9,12,6,13,25,4,15,7,1,3,19],11)
[25, 19, 15, 13, 12, 9, 7, 4, 6, 3, 1]