Christopher AboShear

COSC 3319 01 Data Structures and Algorithms

MWF 9:00 – 9:50am

Successfully Completed Option “A”

**C option input.txt**

11

23

896

473

1547

2096

12

0

-45

1864

78

1024

**C option sorted.txt**

-45

0

12

23

78

473

896

1024

1547

1864

2096

**B option input.txt**

14

February 22 1956

March 24 1856

March 24 1847

December 17 2019

April 1 2019

May 25 2006

November 23 2019

June 12 1862

December 6 2019

January 23 2019

October 22 1956

September 5 1492

July 16 1956

August 17 1956

**B option sorted.txt**

SEPTEMBER 5 1492

MARCH 24 1847

MARCH 24 1856

JUNE 12 1862

FEBRUARY 22 1956

JULY 16 1956

AUGUST 17 1956

OCTOBER 22 1956

MAY 25 2006

JANUARY 23 2019

APRIL 1 2019

NOVEMBER 23 2019

DECEMBER 6 2019

DECEMBER 17 2019

**main.adb**

with Ada.Text\_IO; use Ada.Text\_IO;

procedure Main is

type Month is (January,February,March,April,May,June,July,August,September,October,November,December);

-- record = java class

type Date is record

day : Integer range 1..31;

m : Month;

year : Integer range 0..9999;

end record;

-- function has return. Procedure does not

-- function needs arguments. Procedure does not

-- in = copy. in out = reference ( actual object )

-- Return String of Date

-- Arguments:

-- Date to Print (d)

-- Return:

-- String Representation of Date

function Print\_Date(d : in Date) return String is

begin

-- Image = ToString in java

-- & = concatenation of string

-- Enum'Image you can do. Record'Image you can't

return (Month'Image(d.m) & " " & Integer'Image(d.day) & " " & Integer'Image(d.year));

end Print\_Date;

-- Compares Two Dates

-- Arguments:

-- Two Date Objects to compare

-- Return:

-- True if first date object is greater; false otherwise

function Compare\_Less\_Date(d, d1 : in Date) return Boolean is

begin

if d.year < d1.year then

return True;

elsif d.year = d1.year then

if d.m < d1.m then

return True;

elsif d.m = d1.m then

if d.day < d1.day then

return True;

end if;

end if;

end if;

return False;

end Compare\_Less\_Date;

function Compare\_Greater\_Equal\_Date(d, d1 : in Date) return Boolean is

begin

if d.year >= d1.year then

return True;

elsif d.year = d1.year then

if d.m > d1.m then

return True;

elsif d.m = d1.m then

if d.day > d1.day then

return True;

end if;

end if;

end if;

return False;

end Compare\_Greater\_Equal\_Date;

-- Compare two intergers

-- Arguments:

-- Two Integers

-- Return:

-- True if first integer is greater. False Otherwise

function Compare\_Less\_Integer(i,i1: in Integer) return Boolean is

begin

if i < i1 then

return True;

else

return False;

end if;

end Compare\_Less\_Integer;

function Compare\_Greater\_Equal\_Integer(i,i1: in Integer) return Boolean is

begin

if i >= i1 then

return True;

else

return False;

end if;

end Compare\_Greater\_Equal\_Integer;

-- Type <Type\_Name> is array (<Index\_Data\_Type> range <> ) of <Value\_Data\_Type>;

-- Integer Declaration

type Integer\_Ar is array( Integer range <> ) of Integer;

type Date\_Ar is array( Integer range <> ) of Date;

generic

type Item\_Type is private;

type Array\_Type is array (Integer range <>) of Item\_Type;

with function ">="(i1,i2: in Item\_Type) return Boolean;

with function "<"(i1,i2: in Item\_Type) return Boolean;

procedure Heap\_Sort(Ar: in out Array\_Type; n: in Integer);

-- HeapSort Implementation

-- Arguments:

-- Ar is an array

-- n is the size of the array

procedure Heap\_Sort(Ar : in out Array\_Type; n: in Integer) is

m : Integer := n/2; -- start at middle of tree

r : Integer := n -1;

R1 : Item\_Type;

K : Item\_Type;

j : Integer;

i : Integer;

begin

-- Build initial heap

loop

if m > 0 then

m := m - 1;

R1 := Ar(m);

K := Ar(m);

else

R1 := Ar(r);

K := Ar(r);

Ar(r) := Ar(0);

r := r - 1;

if r = 0 then

Ar(0) := R1; exit;

end if;

end if;

j := m;

loop

i := j;

j := 2\*j + 1;

if j < r then

if Boolean'(Ar(j) < Ar(j+1)) then

j := j + 1;

end if;

if Boolean'(K >= Ar(j)) then

Ar(i) := R1;

exit;

else

Ar(i) := Ar(j);

end if;

elsif (j = r) then

if Boolean'(K >= Ar(j)) then

Ar(i) := R1;

exit;

else

Ar(i) := Ar(j);

end if;

elsif (j > r) then

Ar(i) := R1;

exit;

end if;

end loop;

end loop;

end Heap\_Sort;

generic

type Item\_Type is private;

type Array\_Type is array (Integer range <>) of Item\_Type;

with function image(Item : in Item\_Type) return String;

procedure Print\_Array(Ar: in Array\_Type);

procedure Heap\_Sort\_Integer is new Heap\_Sort(Integer,Integer\_Ar,Compare\_Greater\_Equal\_Integer,Compare\_Less\_Integer);

procedure Heap\_Sort\_Date is new Heap\_Sort(Date,Date\_Ar,Compare\_Greater\_Equal\_Date,Compare\_Less\_Date);

-- Prints array

-- Arguments:

-- Ar is an array

procedure Print\_Array(Ar: in Array\_Type) is

begin

-- loop through each item

for i in Ar'Range

loop

-- print it

Put\_Line(image(Ar(i)));

end loop;

end Print\_Array;

procedure Print\_Array\_Integer is new Print\_Array(Integer,Integer\_Ar,Integer'Image);

procedure Print\_Array\_Date is new Print\_Array(Date,Date\_Ar,Print\_Date);

-- Return substring of string from start\_index to end\_index

-- Arguments:

-- s which is the string to get substring from

-- start\_index which is the index where to start copy from

-- end\_index which is the index where to stop copying

-- Return:

-- substring

function Split\_String(s : in String; start\_index, end\_index : in Integer) return String is

temp : String(1..(end\_index - start\_index + 1)); --holder for the string

counter : Integer := 1; -- holder for start

begin

-- iterate over string from start\_index to end\_index

for i in Integer range start\_index..end\_index

loop

-- copy contents of s(i) to temp

temp(counter) := s(i);

counter:=counter+1;

end loop;

return temp;

end Split\_String;

-- Returns Date Object

-- Arguments:

-- line which is the string to extract date from

-- Return:

-- Date Object

function Get\_Date(line: in String) return Date is

d : Date; -- date object placeholder

start\_index : Integer := line'First;

end\_index : Integer := 0;

counter : Integer := 0; -- keeps track of number of characters

k : Integer := 0; -- flag for month, day, year

begin

-- iterate over string

for i in line'Range

loop

-- if found delimeter

if line(i) = ' ' or counter = line'Length -1 then

-- get end index

end\_index := i - 1;

-- special case for year

if k = 2 then

end\_index := i;

end if;

declare

temp : String := Split\_String(line,start\_index,end\_index); -- month or day or year as string

begin

if k = 0 then -- month

d.m := Month'Value(temp); -- set month in date object

k := k +1;

start\_index := i +1;

elsif k = 1 then

d.day := Integer'Value(temp); -- set day in date object

k := k +1;

start\_index := i +1;

else

d.year := Integer'Value(temp); -- set year in date object

end if;

end;

end if;

counter := counter + 1;

end loop;

return d;

end Get\_Date;

Input, Output : File\_Type;

size : Integer;

begin

Put\_Line("Please enter input file name: ");

declare

input\_file : String := Get\_Line;

begin

Put\_Line("Please enter output file name: ");

declare

output\_file : String := Get\_Line;

begin

Open(File=>Input,

Mode=>Ada.Text\_IO.In\_File,

Name=>input\_file);

Create(File=>Output,

Mode=>Ada.Text\_IO.Out\_File,

Name=>output\_file);

size := Integer'Value(Get\_Line(Input));

declare

--Arr : Integer\_Ar(0..size-1);

Arr\_Date : Date\_Ar(0..size-1);

counter : Integer := 0;

begin

loop

if counter = size then

exit;

end if;

Arr\_Date(counter) := Get\_Date(Get\_Line(input));

--Arr(counter) := Integer'Value(Get\_Line(Input));

counter := counter + 1;

end loop;

Close(Input);

Heap\_Sort\_Date(Arr\_Date,Arr\_Date'Length);

--Heap\_Sort\_Integer(Arr,Arr'Length);

--for i in Arr'Range

for i in Arr\_Date'Range

loop

Put\_Line(Output,Month'Image(Arr\_Date(i).m) & " " & Integer'Image(Arr\_Date(i).day) & " " & Integer'Image(Arr\_Date(i).year));

--Put\_Line(Output,Integer'Image(Arr(i)));

end loop;

Close(Output);

end;

exception

when End\_Error =>

if Is\_Open(Input) then

Close (Input);

end if;

if Is\_Open(Output) then

Close (Output);

end if;

end;

end;

end Main;