Derek Irvin

Professor Ouyang

Assignment 3 Heapsort

**Design:** Heavily Based On Class Notes. Very Simple. See Code Comments

Dalmations = 101 for the size of the array

Node with the nice value ranging from (0 – 39) and sequence value from 1 – 100

Create Node

Assign nice and Sequence values

Insert Heap

Check if the heap is full (size of 101)

Display to the user the heap is full

Else

Go to the next spot in the heap

While hole is greater than position 1 and the insertion nice value is less than its parent

Set the hole position to hole/2

Else set the hole to the niceValue

Deletemin

Check if the size of the heap is equal to 0 (empty)

Display message

Return

Remove the root and PercolateDown by 1

Percolate Down is exact from class notes

printHeap

Loop till all values are displayed nicevalue(sequence Value)

fillHeap

Random Number Generator (0 – 39)

Fill array from 0 – 100

Main

Fill Array

Print Array

Call Heap

Print Heap

Deletemin Heap and fill Heap 2

Print Heap 2

**Output:**

Original Array:

6(1) 10(2) 16(3) 5(4) 28(5) 7(6) 18(7) 5(8) 6(9) 37(10) 4(11) 14(12) 15(13) 37(14) 21(15) 9(16) 22(17) 9(18) 33(19) 35(20) 9(21) 19(22) 28(23) 6(24) 20(25) 22(26) 16(27) 33(28) 33(29) 20(30) 10(31) 11(32) 31(33) 36(34) 8(35) 36(36) 13(37) 9(38) 35(39) 6(40) 6(41) 26(42) 10(43) 6(44) 35(45) 16(46) 3(47) 10(48) 28(49) 10(50) 9(51) 37(52) 2(53) 9(54) 1(55) 28(56) 17(57) 32(58) 30(59) 15(60) 5(61) 37(62) 23(63) 4(64) 24(65) 21(66) 5(67) 23(68) 25(69) 37(70) 32(71) 20(72) 24(73) 31(74) 6(75) 5(76) 0(77) 23(78) 1(79) 12(80) 21(81) 6(82) 29(83) 30(84) 26(85) 25(86) 6(87) 18(88) 27(89) 37(90) 9(91) 29(92) 24(93) 10(94) 24(95) 25(96) 1(97) 14(98) 28(99) 0(100)

Heapsort:

0(77) 1(79) 0(100) 4(64) 3(47) 1(97) 5(61) 5(8) 5(4) 6(40) 4(11) 6(24) 1(55) 17(57) 10(31) 5(67) 8(35) 6(75) 5(76) 6(41) 6(87) 6(44) 10(94) 7(6) 9(51) 2(53) 9(54) 28(56) 30(59) 15(60) 18(7) 9(16) 6(1) 23(68) 22(17) 20(72) 9(18) 9(38) 6(9) 12(80) 6(82) 26(42) 9(21) 18(88) 9(91) 16(46) 24(95) 10(48) 14(98) 10(50) 14(12) 37(52) 22(26) 15(13) 16(27) 37(14) 33(28) 32(58) 33(29) 21(15) 20(30) 37(62) 23(63) 11(32) 24(65) 21(66) 31(33) 36(34) 25(69) 37(70) 32(71) 36(36) 24(73) 31(74) 13(37) 33(19) 10(2) 23(78) 35(39) 35(20) 21(81) 37(10) 29(83) 30(84) 26(85) 25(86) 10(43) 19(22) 27(89) 37(90) 35(45) 29(92) 24(93) 28(23) 28(5) 25(96) 16(3) 28(49) 28(99) 20(25)

DeleteMin:

0(77) 0(100) 1(79) 1(97) 1(55) 2(53) 3(47) 4(64) 4(11) 5(8) 5(67) 5(4) 5(76) 5(61) 6(1) 6(75) 6(9) 6(40) 6(41) 6(82) 6(87) 6(44) 6(24) 7(6) 8(35) 9(16) 9(18) 9(38) 9(21) 9(91) 9(51) 9(54) 10(43) 10(2) 10(94) 10(48) 10(50) 10(31) 11(32) 12(80) 13(37) 14(98) 14(12) 15(13) 15(60) 16(46) 16(3) 16(27) 17(57) 18(88) 18(7) 19(22) 20(72) 20(25) 20(30) 21(66) 21(81) 21(15) 22(17) 22(26) 23(68) 23(78) 23(63) 24(93) 24(65) 24(73) 24(95) 25(69) 25(86) 25(96) 26(85) 26(42) 27(89) 28(99) 28(5) 28(23) 28(56) 28(49) 29(92) 29(83) 30(59) 30(84) 31(33) 31(74) 32(58) 32(71) 33(19) 33(29) 33(28) 35(20) 35(39) 35(45) 36(36) 36(34) 37(52) 37(70) 37(10) 37(90) 37(14) 37(62)

DeleteMin and Heapsort are NOT the same.