HEAPSORT

Sorting an array
Step 1. Think of the array as a binary tree
Step 2. This node has no children, so leave it.
Step 3. This node has no children, so leave it.
Step 4. This node has no children, so leave it
Step 5. This node has no children, so leave it
Step 6. This node has no children, so leave it.
Step 7. This node has no children, so leave it.
Step 8. This node has no children, so leave it.
Step 9. This node has no children, so leave it
Step 10. This node has no children, so leave it
Step 11. This node has no children, so leave it
Step 12. This node has no children, so leave it.
Step 13. This node has no children, so leave it.
Step 14. This node has no children, so leave it.
Step 15. This node has no children, so leave it
Step 16. This node has children
Step 17. Compare the node with its child
Step 17. Its child is larger than it
Step 17. Swap it with its child
Step 17. It has no children
Step 17. So we are done
Step 18. This node has children
Step 19. Compare the node with its two children
Step 19. One of its children is larger
Step 19. Swap it with its largest child
Step 19. It has no children
Step 19. So we are done
Step 20. This node has children
Step 21. Compare the node with its two children
Step 21. One of its children is larger
Step 21. Swap it with its largest child
Step 21. It has no children
HEAPSORT


INPUT

OUTPUT

Step 21. So we are done
Step 22. This node has children
Step 23. Compare the node with its two children
Step 23. One of its children is larger
Step 23. Swap it with its largest child
Step 23. It has no children
Step 23. So we are done
Step 24. This node has children
Step 25. Compare the node with its two children.
Step 25. The elements are in the correct order
Step 25. So we are done
Step 26. This node has children
Step 27. Compare the node with its two children
Step 27. The elements are in the correct order
HEAPSORT


INPUT

ALGORITHMSERT - 2016 - H - APCOAM

OUTPUT

Step 27. So we are done
Step 28. This node has children
Step 29. Compare the node with its two children
HEAPSORT

Step 29. The elements are in the correct order
Step 29. So we are done
Step 30. This node has children
Step 31. Compare the node with its two children
HEAPSORT

Step 31. The elements are in the correct order
HEAPSORT

Step 31. So we are done
Step 32. This node has children
Step 33. Compare the node with its two children
Step 33. One of its children is larger
Step 33. Swap it with its largest child
Input

Output

Step 34. Compare the node with its two children
Step 34. One of its children is larger
Step 34. Swap it with its largest child
HEAPSORT

Step 34. It has no children
Step 34. So we are done
Step 35. This node has children
Step 36. Compare the node with its two children
Step 36. One of its children is larger
Step 36. Swap it with its largest child
Step 37. Compare the node with its two children.
Step 37. The elements are in the correct order
Step 37. So we are done
Step 38. This node has children
Step 39. The elements are in the correct order
Step 39. So we are done.
Step 40. This node has children
Step 41. Compare the node with its two children
Step 41. One of its children is larger
HEAPSORT

Step 41. Swap it with its largest child
Step 42. Compare the node with its two children
Step 42. One of its children is larger
HEAPSORT

Step 42. Swap it with its largest child
Step 43. Compare the node with its child
Step 43. Its child is larger than it
HEAPSORT

Step 43. Swap it with its child
Step 43. It has no children
HEAPSORT

Step 43. So we are done
HEAPSORT


Step 44. This node has children
Step 45. Compare the node with its two children
Step 45. One of its children is larger
Step 45. Swap it with its largest child
Step 46. Compare the node with its two children
Step 46. One of its children is larger
Step 46. Swap it with its largest child
Step 47. Compare the node with its two children.
Step 47. The elements are in the correct order.
HEAPSORT

Step 47. So we are done
Step 48. This node has children
Step 49. Compare the node with its two children
Step 49. One of its children is larger
Step 49. Swap it with its largest child
HEAPSORT

Step 50. Compare the node with its two children
Step 50. One of its children is larger
Step 50. Swap it with its largest child
Step 51. Compare the node with its two children
Step 51. One of its children is larger
HEAPSORT

Step 51. Swap it with its largest child
Step 52. Compare the node with its child
HEAPSORT

Step 52. Its child is larger than it
Step 52. Swap it with its child
Step 52. It has no children
HEAPSORT

Step 52. So we are done
Step 53. The tree is now a max-heap
Step 54. We now keep taking the root element
HEAPSORT

Removing the root
Step 1. Find the root of the heap
Step 2. Output the value of the root
Step 3. Find the last node
HEAPSORT

Step 4. Move the last node to the root
Step 5. Compare the node with its two children.
HEAPSORT

Step 6. One of its children is larger
HEAPSORT

Step 6. Swap it with its largest child
Step 6. Compare the node with its two children
Step 7. One of its children is larger
Step 7. Swap it with its largest child
Step 7. Compare the node with its two children
Step 8. One of its children is larger
HEAPSORT

Step 8. Swap it with its largest child
HEAPSORT

Step 8. Compare the node with its two children
Step 9. One of its children is larger
Step 9. Swap it with its largest child
Step 9. It has no children
Step 9. So we are done
HEAPSORT

INPUT

Removing the root

OUTPUT
HEAPSORT

Step 1. Find the root of the heap
HEAPSORT

Step 2. Output the value of the root
HEAPSORT

Step 3. Find the last node

INPUT


OUTPUT

T, T
HEAPSORT

Step 4. Move the last node to the root
Step 5. Compare the node with its two children
Step 6. One of its children is larger
Step 6. Swap it with its largest child
Step 6. Compare the node with its two children
HEAPSORT

Step 7. One of its children is larger
Step 7. Swap it with its largest child
HEAPSORT

Step 7. Compare the node with its two children
HEAPSORT

Step 8. One of its children is larger
HEAPSORT

Step 8. Swap it with its largest child
Step 8. Compare the node with its two children
Step 9. One of its children is larger
Step 9. Swap it with its largest child
Step 9. It has no children
HEAPSORT

Step 9. So we are done
HEAPSORT

Removing the root
Step 1. Find the root of the heap
HEAPSORT

Step 2. Output the value of the root

INPUT

OUTPUT


S

T, T

RSOLRMHMMHEPG-2016-A-AICA
Step 3. Find the last node
Step 4. Move the last node to the root
Step 5. Compare the node with its two children
HEAPSORT

Step 6. One of its children is larger
Step 6. Swap it with its largest child
Step 6. Compare the node with its two children
Step 7. One of its children is larger
Step 7. Swap it with its largest child
Step 7. Compare the node with its two children
Step 8. One of its children is larger
Step 8. Swap it with its largest child.
Step 8. Compare the node with its two children
Step 9. One of its children is larger
Step 9. Swap it with its largest child
Step 9. So we are done
HEAPSORT

Removing the root
HEAPSORT


INPUT

Step 1. Find the root of the heap

OUTPUT

S, T, T

SRROLPMHMHHEIQG-2015-A-AAC
Step 2. Output the value of the root
Step 3. Find the last node
Step 4. Move the last node to the root
Step 5. Compare the node with its two children
Step 6. One of its children is larger
Step 6. Swap it with its largest child
Step 6. Compare the node with its two children
HEAPSORT

Step 7. One of its children is larger
Step 7. Swap it with its largest child
Step 7. Compare the node with its two children.
Step 8. One of its children is larger
Step 8. Swap it with its largest child
Step 8. Compare the node with its two children
Step 9. The elements are in the correct order
HEAPSORT

Step 9. So we are done
HEAPSORT

INPUT

Removing the root

OUTPUT
HEAPSORT

Step 1. Find the root of the heap
HEAPSORT

Step 2. Output the value of the root
Step 3. Find the last node
Step 4. Move the last node to the root
Step 5. Compare the node with its two children
HEAPSORT

Step 6. One of its children is larger
Step 6. Swap it with its largest child
HEAPSORT

Step 6. Compare the node with its two children
Step 7. One of its children is larger
Step 7. Swap it with its largest child
Step 7. Compare the node with its two children
HEAPSORT

Step 8. Swap it with its largest child
Step 8. It has no children
HEAPSORT

Step 8. So we are done
HEAPSORT

Removing the root
HEAPSORT

Step 1. Find the root of the heap


INPUT

OUTPUT

R, S, S, T, T

ROPMLDMMHCEHAG-2015-A-A
Step 2. Output the value of the root
HEAPSORT

Step 3. Find the last node

INPUT

A, L, G, O,
R, I, T, H,
M, S, -, C,
A, M, -, 2,
0, 1, 5, -,
H, E, A, P,
S, O, R, T

OUTPUT

R, R,
S, S,
T, T
HEAPSORT

Step 4. Move the last node to the root
Step 5. Compare the node with its two children
Step 6. One of its children is larger
Step 6. Swap it with its largest child
Step 6. Compare the node with its two children
Step 7. One of its children is larger
Step 7. Swap it with its largest child
Step 7. Compare the node with its two children
HEAPSORT

Step 8. One of its children is larger
Step 8. Swap it with its largest child
Step 8. It has no children
HEAPSORT

Step 8. So we are done
HEAPSORT

Removing the root
HEAPSORT

Step 1. Find the root of the heap

INPUT

A, L, G, O,
R, I, T, H,
M, S, -, C,
A, M, -, 2,
0, 1, 5, -,
H, E, A, P,
S, O, R, T

OUTPUT

R, R,
S, S,
T, T
Step 2. Output the value of the root
Step 3. Find the last node
HEAPSORT

Step 4. Move the last node to the root
Step 5. Compare the node with its two children
HEAPSORT

Step 6. One of its children is larger
Step 6. Swap it with its largest child
Step 6. Compare the node with its two children
Step 7. One of its children is larger.
Step 7. Swap it with its largest child
Step 7. Compare the node with its two children
Step 8. One of its children is larger
Step 8. Swap it with its largest child
Step 8. Compare the node with its two children.
HEAPSORT

Step 9. One of its children is larger
Step 9. Swap it with its largest child
Step 9. It has no children
Step 9. So we are done
HEAPSORT

Removing the root
Step 1. Find the root of the heap
Step 2. Output the value of the root
Step 3. Find the last node
Step 4. Move the last node to the root
HEAPSORT

Step 5. Compare the node with its two children

INPUT

OUTPUT


O, P, R, R, S, S, T, T
Step 6. One of its children is larger
Step 6. Swap it with its largest child
Step 6. Compare the node with its two children
Step 7. One of its children is larger
Step 7. Swap it with its largest child
Step 7. Compare the node with its two children
HEAPSORT

Step 8. One of its children is larger
Step 8. Swap it with its largest child
HEAPSORT

INPUT

OUTPUT
O, P, R, R, S, S, T, T

Step 8. It has no children
HEAPSORT

Step 8. So we are done
HEAPSORT

Removing the root

INPUT


OUTPUT

O, P, R, R, S, S, T, T
Step 1. Find the root of the heap
Step 2. Output the value of the root
HEAPSORT

INPUT


OUTPUT


Step 3. Find the last node
HEAPSORT

Step 4. Move the last node to the root
Step 5. Compare the node with its two children.
Step 6. One of its children is larger
HEAPSORT

Step 6. Swap it with its largest child
Step 6. Compare the node with its two children
Step 7. One of its children is larger
Step 7. Swap it with its largest child
Step 7. Compare the node with its two children
HEAPSORT

Step 8. One of its children is larger
Step 8. Swap it with its largest child
Step 8. It has no children
HEAPSORT

Step 8. So we are done
HEAPSORT

Removing the root
Step 1. Find the root of the heap
Step 2. Output the value of the root
HEAPSORT

Step 3. Find the last node

INPUT
A, L, G, O, R, I, T, H,
M, S, -, C,
A, M, -, 2,
0, 1, 5, -,
H, E, A, P,
S, O, R, T

OUTPUT
M, O,
O, P,
R, R,
S, S,
T, T
Step 4. Move the last node to the root
Step 5. Compare the node with its two children
Step 6. One of its children is larger
Step 6. Swap it with its largest child
Step 6. Compare the node with its two children
Step 7. One of its children is larger
Step 7. Swap it with its largest child
Step 7. Compare the node with its two children
HEAPSORT

Step 8. One of its children is larger
Step 8. Swap it with its largest child
Step 8. It has no children
Step 8. So we are done
HEAPSORT

Removing the root

INPUT


OUTPUT

Step 1. Find the root of the heap
Step 2. Output the value of the root
Step 3. Find the last node
Step 4. Move the last node to the root
Step 5. Compare the node with its two children.
Step 6. One of its children is larger
Step 6. Swap it with its largest child
Step 6. Compare the node with its two children.
Step 7. One of its children is larger
HEAPSORT

Step 7. Swap it with its largest child
Step 7. Compare the node with its two children
HEAPSORT

Step 8. One of its children is larger
HEAPSORT

Step 8. Swap it with its largest child
HEAPSORT

Step 8. It has no children
HEAPSORT

Step 8. So we are done
HEAPSORT

Removing the root

INPUT


OUTPUT

Step 1. Find the root of the heap
Step 2. Output the value of the root
Step 3. Find the last node
Step 4. Move the last node to the root
Step 5. Compare the node with its two children
HEAPSORT


INPUT

Step 6. One of its children is larger

OUTPUT

Step 6. Swap it with its largest child
Step 6. Compare the node with its two children
Step 7. One of its children is larger
Step 7. Swap it with its largest child
Step 7. Compare the node with its two children
HEAPSORT

Step 8. One of its children is larger

INPUT


OUTPUT

Step 8. Swap it with its largest child
HEAPSORT

INPUT


OUTPUT


Step 8. So we are done
HEAPSORT

INPUT


Removing the root

OUTPUT

Step 1. Find the root of the heap
Step 2. Output the value of the root
Step 3. Find the last node
Step 4. Move the last node to the root
Step 5. Compare the node with its two children
Step 6. One of its children is larger
HEAPSORT

Step 6. Swap it with its largest child
Step 6. Compare the node with its two children
Step 7. One of its children is larger
Step 7. Swap it with its largest child
Step 7. Compare the node with its two children
Step 8. One of its children is larger
Step 8. Swap it with its largest child
Step 8. It has no children
Step 8. So we are done
HEAPSORT

Removing the root

INPUT

OUTPUT
Step 1. Find the root of the heap
Step 3. Find the last node
Step 4. Move the last node to the root
HEAPSORT

Step 5. Compare the node with its two children
Step 6. One of its children is larger
Step 6. Swap it with its largest child
Step 6. Compare the node with its two children
Step 7. One of its children is larger
HEAPSORT

Step 7. Swap it with its largest child
HEAPSORT

Step 7. Compare the node with its two children
Step 8. The elements are in the correct order
Step 8. So we are done
HEAPSORT

Removing the root

INPUT


OUTPUT

Heapsort

Step 1. Find the root of the heap
Step 2. Output the value of the root
Step 3. Find the last node
Step 4. Move the last node to the root
Step 5. Compare the node with its two children
Step 6. One of its children is larger
Step 6. Swap it with its largest child
Step 6. Compare the node with its two children
Step 7. One of its children is larger
HEAPSORT

Step 7. Swap it with its largest child
Step 7. Compare the node with its two children
Step 8. One of its children is larger
HEAPSORT

Step 8. Swap it with its largest child


INPUT

OUTPUT

Step 8. It has no children
Step 8. So we are done
HEAPSORT

INPUT

Removing the root

OUTPUT

A, L, G, O,
R, I, T, H,
M, S, -, C,
A, M, -, 2,
0, 1, 5, -, 
H, E, A, P,
S, O, R, T

H, H,
I, L,
M, M,
O, O,
P, R,
R, S,
S, T,
T
HEAPSORT

Step 1. Find the root of the heap
Step 2. Output the value of the root
Step 3. Find the last node
Step 4. Move the last node to the root
Step 5. Compare the node with its two children
HEAPSORT

Step 6. One of its children is larger
HEAPSORT

Step 6. Swap it with its largest child

INPUT


OUTPUT

Step 6. Compare the node with its two children
Step 7. One of its children is larger
HEAPSORT

Step 7. Swap it with its largest child
HEAPSORT

Step 7. Compare the node with its two children.
Step 8. One of its children is larger
Step 8. Swap it with its largest child
Step 8. It has no children
Step 8. So we are done
HEAPSORT

Removing the root

INPUT


OUTPUT

HEAPSORT

Step 1. Find the root of the heap
Step 2. Output the value of the root
Step 3. Find the last node
HEAPSORT

Step 4. Move the last node to the root
Step 5. Compare the node with its two children.
Step 6. One of its children is larger.
HEAPSORT

Step 6. Swap it with its largest child
Step 6. Compare the node with its two children
Step 7. The elements are in the correct order
Step 7. So we are done
HEAPSORT

Removing the root


Step 1. Find the root of the heap
HEAPSORT

Step 2. Output the value of the root
Step 3. Find the last node
HEAPSORT

Step 4. Move the last node to the root
Step 5. Compare the node with its two children
Step 6. One of its children is larger
Step 6. Swap it with its largest child
Step 6. Compare the node with its two children
HEAPSORT


Step 7. One of its children is larger

Step 7. Swap it with its largest child
Step 7. It has no children
Step 7. So we are done
HEAPSORT

Removing the root

INPUT

A, L, G, O,
R, I, T, H,
M, S, -, C,
A, M, -, 2,
0, 1, 5, -,
H, E, A, P,
S, O, R, T

OUTPUT

C, E,
G, H,
H, I,
L, M,
M, O,
O, P,
R, R,
S, S,
T, T
HEAPSORT

INPUT

OUTPUT


A

Step 1. Find the root of the heap

Step 2. Output the value of the root
Step 3. Find the last node
Step 4. Move the last node to the root
Step 5. Compare the node with its two children
Step 6. One of its children is larger
Step 6. Swap it with its largest child
HEAPSORT

INPUT


OUTPUT


Step 6. Compare the node with its two children
Step 7. One of its children is larger
HEAPSORT

Step 7. Swap it with its largest child
Step 7. It has no children
Step 7. So we are done
Removing the root
HEAPSORT

A, L, G, O,
R, I, T, H,
M, S, C,
A, M, 2,
0, 1, 5,
H, E, A, P,
S, O, R, T

INPUT

OUTPUT

Step 1. Find the root of the heap
Step 2. Output the value of the root
Step 3. Find the last node
Step 4. Move the last node to the root
HEAPSORT

Step 5. Compare the node with its two children.
Step 6. One of its children is larger
Step 6. Swap it with its largest child
HEAPSORT

Step 6. Compare the node with its two children
Step 7. The elements are in the correct order
HEAPSORT

Step 7. So we are done
HEAPSORT

INPUT


OUTPUT


Removing the root
HEAPSORT

Step 1. Find the root of the heap


HEAPSORT

Step 2. Output the value of the root


Step 3. Find the last node
Step 4. Move the last node to the root
HEAPSORT

Step 5. Compare the node with its two children
Step 6. One of its children is larger
HEAPSORT

Step 6. Swap it with its largest child
HEAPSORT

Step 6. Compare the node with its two children

INPUT

A, L, G, O,
R, I, T, H,
M, S, -, C,
A, M, -, 2,
0, 1, 5, -, H, E, A, P,
S, O, R, T

OUTPUT

A, A,
A, C,
E, G,
H, H,
I, L,
M, M,
O, O,
P, R,
R, S,
S, T,
T
HEAPSORT

Step 7. One of its children is larger
HEAPSORT

Step 7. Swap it with its largest child
HEAPSORT

Step 7. It has no children
Step 7. So we are done
HEAPSORT

INPUT

OUTPUT

Removing the root
Step 1. Find the root of the heap
HEAPSORT

Step 2. Output the value of the root
Step 3. Find the last node
HEAPSORT

INPUT


OUTPUT


Step 4. Move the last node to the root
Step 5. Compare the node with its two children
HEAPSORT

Step 6. One of its children is larger
HEAPSORT

Step 6. Swap it with its largest child
Step 6. Compare the node with its two children.
Step 7. One of its children is larger
Step 7. Swap it with its largest child
Step 7. It has no children
HEAPSORT

INPUT


OUTPUT


Step 7. So we are done
HEAPSORT

INPUT


Removing the root

OUTPUT

Step 1. Find the root of the heap
Step 2. Output the value of the root
Step 4. Move the last node to the root
Step 5. Compare the node with its two children
Step 6. One of its children is larger
HEAPSORT

Step 6. Swap it with its largest child
Step 6. It has no children
HEAPSORT

Step 6. So we are done
Removing the root
Step 1. Find the root of the heap
Step 2. Output the value of the root
HEAPSORT

Step 3. Find the last node

INPUT

OUTPUT
Step 4. Move the last node to the root
Step 5. Compare the node with its two children
Step 6. One of its children is larger
Step 6. Swap it with its largest child
Step 6. Compare the node with its child
Step 7. The elements are in the correct order.
Step 7. So we are done
HEAPSORT

INPUT

Removing the root

OUTPUT


Step 1. Find the root of the heap
HEAPSORT

Step 2. Output the value of the root
Step 3. Find the last node
Step 4. Move the last node to the root
HEAPSORT

Step 5. Compare the node with its two children
Step 6. The elements are in the correct order.
Step 6. So we are done
HEAPSORT

INPUT


Removing the root

OUTPUT

Step 1. Find the root of the heap
HEAPSORT

Step 2. Output the value of the root
Step 3. Find the last node
Step 4. Move the last node to the root
HEAPSORT

Step 5. Compare the node with its child
HEAPSORT

Step 6. The elements are in the correct order
HEAPSORT

Step 6. So we are done
Removing the root
Step 1. Find the root of the heap
Step 2. Output the value of the root
Step 3. Find the last node
Step 4. Move the last node to the root
Step 5. The heap has a single element
HEAPSORT

Removing the root
HEAPSORT

Step 1. Find the root of the heap
Step 2. Output the value of the root
Step 3. The heap is now empty
Step 4. So we are done
Step 55. The array has now been sorted