CIS2520 Data Structures
Assignment 4

Dr. Charlie Obimbo, Modified by Yukun **Due: November 26, 2022**

1 Heaps

1. (100%) Write a C program. It reads 200 2-digit integers from text file "f.dat" and stores the integers in an array of 20 rows and 10 columns. The program treats each row of the array as an object, with the sum of the first three integers being the key, and the other seven integers being the information content. The program then creates a heap with a node containing an object. You are required to use an array representation of heap, and apply the parental node downheap algorithm in the array representation. The program finally displays the heap as a 20×10 array, a row for an object.

2 Assignment 4 Guidelines

Assignment 4 is due on Nov 26, 2022.

- Write a makefile that can be used to generate executables, same as the one for Assignment 2.
- The information in *readme* file is the same as before. Other requirements are also the same.
- The file name is **f.dat** (can hard coding) and it contains exactly 200 2-digit integers (20 rows and 10 columns).

Hints:

- The 2D array can be considered as an extension of the 1D array for representing heaps.
- The heap will contain a total of 20 objects.
- Each object holds a row of 10 integers.
- The first three integers are used to derive the key. The key is the sum of the first three integers.
- The last seven integers are the content of the object. For example:

```
typedef struct Node{
  int sum_key;
  int key[3];
  int content[7];
} node;
```

Due time: 11:59 pm Nov 26, 2022.

3 Test case

```
f.dat:
25 00 54 25 19 25 81 21 02 78
19 09 89 77 80 75 91 34 61 24
36 42 65 18 81 93 72 34 59 37
10 56 17 24 47 02 35 45 19 51
26 28 10 23 03 32 65 61 28 95
66 63 94 42 77 64 56 80 63 14
77 07 34 93 04 19 10 44 76 19
86 18 40 47 13 94 98 22 79 94
68 37 41 12 06 85 51 85 60 56
03 98 29 05 60 15 98 86 04 61
77 51 28 24 77 02 36 64 32 05
30 73 12 75 14 85 72 60 91 42
83 46 03 67 90 48 04 74 41 52
62 30 46 71 41 38 80 60 99 05
19 48 83 98 11 30 41 72 09 31
31 44 21 79 68 97 32 13 62 80
61 69 82 25 62 12 83 35 56 19
62 74 67 19 41 35 38 16 09 80
47 44 85 30 84 53 28 42 07 65
99 30 00 91 26 09 91 70 21 14

>./A4
26 28 10 23 03 32 65 61 28 95
25 00 54 25 19 25 81 21 02 78
30 73 12 75 14 85 72 60 91 42
10 56 17 24 47 02 35 45 19 51
19 09 89 77 80 75 91 34 61 24
83 46 03 67 90 48 04 74 41 52
77 07 34 93 04 19 10 44 76 19
31 44 21 79 68 97 32 13 62 80
68 37 41 12 06 85 51 85 60 56
99 30 00 91 26 09 91 70 21 14
77 51 28 24 77 02 36 64 32 05
66 63 94 42 77 64 56 80 63 14
36 42 65 18 81 93 72 34 59 37
62 30 46 71 41 38 80 60 99 05
19 48 83 98 11 30 41 72 09 31
86 18 40 47 13 94 98 22 79 94
61 69 82 25 62 12 83 35 56 19
62 74 67 19 41 35 38 16 09 80
47 44 85 30 84 53 28 42 07 65
03 98 29 05 60 15 98 86 04 61
```

Explanation:

```
In test case f.dat:

Node.sum_key: 79 (25+0+54)
Node.sum_key: 117 (19+9+89)
Node.sum_key: 143 (36+42+65)
Node.sum_key: 83 (10+56+17)
Node.sum_key: 64 (26+28+10)
Node.sum_key: 223 (66+63+94)
Node.sum_key: 118 (77+7+34)
Node.sum_key: 144 (86+18+40)
Node.sum_key: 146 (68+37+41)
Node.sum_key: 130 (3+98+29)
Node.sum_key: 156 (77+51+28)
Node.sum_key: 115 (30+73+12)
Node.sum_key: 132 (83+46+3)
Node.sum_key: 138 (62+30+46)
Node.sum_key: 150 (19+48+83)
Node.sum_key: 96 (31+44+21)
Node.sum_key: 212 (61+69+82)
Node.sum_key: 203 (62+74+67)
Node.sum_key: 176 (47+44+85)
Node.sum_key: 129 (99+30+0)

After apply the parental node downheap algorithm:

Node.sum_key after downheap: 64 (26+28+10)
Node.sum_key after downheap: 79 (25+0+54)
Node.sum_key after downheap: 115 (30+73+12)
Node.sum_key after downheap: 83 (10+56+17)
Node.sum_key after downheap: 117 (19+9+89)
Node.sum_key after downheap: 132 (83+46+3)
Node.sum_key after downheap: 118 (77+7+34)
Node.sum_key after downheap: 96 (31+44+21)
Node.sum_key after downheap: 146 (68+37+41)
Node.sum_key after downheap: 129 (99+30+0)
Node.sum_key after downheap: 156 (77+51+28)
Node.sum_key after downheap: 223 (66+63+94)
Node.sum_key after downheap: 143 (36+42+65)
Node.sum_key after downheap: 138 (62+30+46)
Node.sum_key after downheap: 150 (19+48+83)
Node.sum_key after downheap: 144 (86+18+40)
Node.sum_key after downheap: 212 (61+69+82)
Node.sum_key after downheap: 203 (62+74+67)
Node.sum_key after downheap: 176 (47+44+85)
Node.sum_key after downheap: 130 (3+98+29)
```