CSPB 2270 - Summer 2023 Jones - Data Structures & Algorithms

Dashboard / My courses / 2234:CSPB 2270 / 31 July - 6 August / Exam 2 (Remotely Proctored)

Started on	Friday, 4 August 2023, 1:49 PM
State	Finished
Completed on	Friday, 4 August 2023, 2:54 PM
Time taken	1 hour 4 mins
Grade	**85.25** out of 100.00

Information

For the following problems, here are the definitions of two recursive functions:

```
void thing(int v) {
  cout << v << ",";
  if (v < 0) {
    return;
  }
  if (v % 2 == 0) {
    thing(v - 3);
  } else {
    thing(v - 1);
  }
}

void otherthing(int v) {
  if (v < 0) {
    return;
  }
  if (v % 2 == 0) {
    thing(v - 3);
  } else {
    thing(v - 1);
  }
  cout << v << ",";
}
```

Question **1**

Complete

Mark 0.00 out of 0.50

What is the exact output when `thing(5)` is called?

Answer: 5,4,1,0,-1,

Correct answer is

5,4,1,0,-3,

Question **2**

Complete

Mark 0.00 out of 0.50

What is the output when `otherthing(5)` is called?

Answer: 4,1,0,-1,

Correct answer is

4,1,0,-3,5,

Information

Say we have the following structure definition:

```
```

```
struct goo {
  shared_ptr<goo> a;
  shared_ptr<shared_ptr<goo>> b;
};
```

Question **3**

Complete

Mark 15.00 out of 15.00

Allocate two new variables. The idea here is to show that you know how to declare and assign variables and know the difference between heap and stack memory.

1. Variable x is a 'pointer to goo', which is stored on heap memory. Declare and assign it.
   - shared_ptr<goo> x(new goo);
2. Variable x is a 'goo' value (not a pointer), and is stored with stack memory. Simply declare it (no need to assign it a value).
   - goo x;

Note: Drag and drop the correct answer from choices bellow.

goo x = shared_pr<goo>(new goo);

shared_ptr<goo>x; shared_ptr<x> goo; goo x = new goo*;

Your answer is correct.

Information

Here is a simplistic hash function used to look up the bucket address within a hash table.

```
int h(int v, int n) {
  return v % n;
}
```

Question **4**

Complete

Mark 5.00 out of 5.00

How many unique buckets can this hash function address?

Answer: n

Well done, Your answer is correct.

Correct answer is : n

Question **5**

Complete

Mark 15.00 out of 15.00

Say we are using linear probing, and have the following hash table:

```
bucket -> value
  0    ->  --
  1    ->  9
  2    ->  18
  3    ->  3
  4    ->  12
  5    ->  --
  6    ->  14
  7    ->  --
```

(So for example, the value in bucket 1 is 9)

The value `20` is to be inserted into this hashtable using our `h` bucket hash function defined earlier. What does the table look like when that operation is finished?

Note that you should be able to figure out what the value of `n` is by looking at the bucket list.

(Note: please place empty options in order)

bucket	->	value
0	->	empty-first
1	->	9
2	->	18
3	->	3
4	->	12
5	->	20
6	->	14
7	->	empty-second

Your answer is correct.

correct answer is

bucket:	0	1	2	3	4	5	6	7
value :	--	9	18	3	12	20	14	--

Question **6**

Complete

Mark 10.00 out of 10.00

Using your table with the `20` inserted, now write out what the table looks like when `11` is inserted.

bucket	-> value
0	--
1	9
2	18
3	3
4	12
5	20
6	14
7	11

Your answer is correct.

Correct answer is :

bucket:	0	1	2	3	4	5	6	7
value :	--	9	18	3	12	20	14	11

Question **7**

Complete

Mark 5.00 out of 5.00

Given this hash function, what is the hashed value for `417`?

```
int porcupine(int n) {
    return ((n + 4) % 8);
}
```

Answer: 5

Question **8**

Complete

Mark 5.00 out of 5.00

Say we have a b-tree (`max_children = 5`) node with keys in the order [5, 10, 16, 22]. Claim: this node violates invariants related to keys.

Select one:

- True
- [✓ False]

Information

What is the *runtime complexity* of the following operations?

Question **9**

Complete

Mark 3.00 out of 3.00

Insert a value into a min heap?

Select one:

- a. O(1)
- [✓ b. O(log n)]
- c. O(n)
- d. O(n log n)
- e. O(n^2)

Your answer is correct.

Question **10**

Complete

Mark 3.00 out of 3.00

Reading the smallest value in a min heap?

Select one:

[✓ a. O(1)]

b. O(log n)

c. O(n)

d. O(n log n)

e. O(n^2)

Your answer is correct.

Question **11**

Complete

Mark 3.00 out of 3.00

Removing an item from a B-tree?

Select one:

a. O(1)

[✓ b. O(log n)]

c. O(n)

d. O(n log n)

e. O(n^2)

Your answer is correct.

Question **12**

Complete

Mark 10.00 out of 10.00

Say we create an empty priority queue that prioritizes large values over smaller ones. We add and remove things from it as follows. `pop()` removes and returns the highest priority value, `peek()` returns (but does *not remove it*), and `add(v)` adds the value `v` to the queue at the appropriate location.

Write the contents of the queue in their priority order after this sequence of operations.

```
Q = (empty priority queue)
Q.add(14)
Q.add(20)
Q.add(8)
Q.pop()
Q.peek()
Q.add(17)
Q.add(9)
Q.pop()
Q.pop()
```

Answer style: enter your answer values in comma separated format with higher priority values to the left. for example

11,10,9

Answer: 9,8

Well done, your answer is correct.

Correct answer is : 9,8

Information

This is a sequence of cumulative questions. You have to get the previous one right in order to get points for later questions.

Huffman encoding uses a priority queue to build an encoding tree based on frequency information per symbol, where low frequencies have higher priority. Supernodes combine lower-level nodes, and their frequency is the the sum of the child nodes' frequencies.

The algorithm could look like this:

```
pq = (priority queue with leaf nodes for every symbol)
while (pq has more than one element):
    a = pq.pop()   -- this removes the highest priority item and returns it
    b = pq.pop()
    c = build_supernode(a, b) -- c.left is a, c.right is b
    pq.insert(c)
tree = pq.pop() -- top of encoding tree
```

Say we have a small symbol set comprising the letters A, O, M, and N. Also say our initial priority queue looks like the following (each nodes has the symbol followed by its frequency).

```
[N: 3], [O: 4], [A: 11], [M: 8]
```

Question **13**

Complete

Mark 10.00 out of 10.00

If you read the description closely you might have spotted a bug in the ordering of nodes in the initial priority queue.

You can fix this bug by swapping two elements. What would be the state of the priority queue after this swap?

Answer: [N: 3] , [O: 4] , [M: 8] , [A: 11]

None

Your answer is correct.

Question **14**

Complete

Mark 1.25 out of 5.00

Now that you've hopefully fixed the initial priority queue, your next task is to build an encoding lookup table. Execute the above tree-building algorithm using your corrected ordering of the priority queue. Read the comment next to `build_supernode`, because that is what guarantees that the answer is unique. Assuming `0 = left, 1 = right`, write the correct bitstring associated with each symbol in our little dictionary.

fill in the following bit strings (you should only have 0 or 1 characters in your strings)

A:
0

O:
110

M:
10

N:
111

Your answer is partially correct.

1 of your answers is correct.

Question **15**

Complete

Mark 0.00 out of 5.00

Using the symbol map you just wrote out, decode the following bit string back into English. The result is legible, if you insert spaces, but leave spaces out in your answer.
`10010111010010110011101101100`

Answer: MAMNAMMAOANAOOA

Question **16**

Complete

Mark 0.00 out of 5.00

Using the same symbol map, what is the bitstring encoding for `OMA`?

Answer: 110100