## Study for Examination March 6, 2024

1. Fill in the blanks.

   (a) Any comparison-based sorting algorithm on a file of size $n$ must execute ______________ comparisons in the worst case. Use $\Omega$.

   (b) Name two well-known divide-and-conquer sorting algorithms.

   ____________________________

   ____________________________

2. Fill in each blank. Write $\Theta$ if that is correct; otherwise write $O$ or $\Omega$, whichever is correct. Recall that log means $\log_2$.

   (a) $\log n^2 =$ _______ $\log n^3$

   (b) $\log(n!) =$ _______ $n \log n$

   (c) $\sum_{i=0}^{n-1} i^k =$ _______ $n^k$

   (d) $n^n =$ _______ $2^{\log^2 n}$.

   (e) $\log n =$ _______ $\ln n$

3. Fill in each blank with one of the words, *stack*, *heap*, *queue*, or *array*.

   (a) "pop" and "push" are operators of ______________.

   (b) "fetch" and "store" are operators of ______________.

4. Find the asymptotic time complexity of each of these code fragments in terms of $n$, using $\Theta$ notation.

   (a)
   ```
   for(int i = 0; i*i < n; i++)
   ```

   (b)
   ```
   for(int i = 0; i < n; i++)
     for(int j = 1; j < i; j = 2*j);
   ```

   (c)
   ```
   for(int i = 1; i < n; i++)
     for(int j = i; j < n; j = 2*j);
   ```

   (d)
   ```
   for(float x = n; x > 2.0; x = sqrt(x))
   ```
   (`sqrt(x)` returns the square root of `x`.)

   (e)
   ```
   for(int i = 1; i < n; i = 2*i)
     for(int j = 2; j < i; j = j*j);
   ```
   (Hint: use substitution)

5. Show a circular queue with dummy node items B, M, Q, R, in that order, from front to rear. then show how the queue changes when you insert H.

6. A stack of integers could be implemented in C++ as a linked list as follows.

```
struct stack
 {
  int item;
  stack*link;
 };
```

Finish writing the code for the operators push, pop, and empty, below.

```
void push(stack*&s,int newitem)
```

```
int pop(stack*&s)
```

```
bool empty(stack*s)
```

7. Let $F_1, F_2, \ldots$ be the Fibonacci numbers. Find a constant $K$ such that $F_n = \Theta(K^n)$. Show the steps.

8. (a) What is the purpose of the function `power` given below?

   (b) Find a loop invariant of the while loop.

```
float power(float x, int n) // input condition: n >= 0
 {
  int m = n;
  float y = x;
  float z = 1.0;
  while(m > 0)
   {
    if(m%2) z = z*y;
    m = m/2;
    y = y*y;
   }
  return z;
 }
```

9. The following code could be used in a C++ program implementing quicksort. What is the loop invariant of the loop?

```
void quicksort(int first,int last) // sorts the subarray A[first .. last]
 {
  if(first < last) // if first >= last, we are done
   {
    int mid = (first+last)/2;
    int pivot = A[mid];
    swap(A[mid],A[first]); // move pivot to first position
    int lo = first;
    int hi = last;
    while(lo < hi)
     {
      if(A[lo+1] > pivot and A[hi] < pivot)
       {
        swap(A[lo+1],A[hi]);
        lo++;
        hi--;
       }
      if(A[lo+1] <= pivot and lo < hi) lo++;
      if(A[hi] >= pivot and lo < hi) hi--;
     }
    assert(lo == hi);
    swap(A[first],A[lo]); // move pivot between subarrays
    if(lo < mid) // sort the smaller subarray first
     {
      quicksort(first,lo-1);
      quicksort(lo+1,last);
     }
    else
     {
      quicksort(lo+1,last);
      quicksort(first,lo-1);
     }
   }
 }
```

10. The following portion of C++ code contains an array implementation of `queue`. Fill in the missing code for the operators "`enqueue`" and "`empty`."

```
struct queue
 {
  int A[N]; // N is a constant large enough to prevent overflow
  int rear = 0;
  int front = 0; // initially the queue is empty
 };

void enqueue(queue&q,int newitem) // inserts newitem into q
 {
 }

bool empty(queue q) // returns true if q is empty, false otherwise
 {
 }

int dequeue(queue&q) // returns an item from q and deletes that item
 {
  int rslt = q.A[q.front];
  q.front++;
  return rslt;
 }
```

11. Fill in the blanks.

    (a) Name three search structures.

    --------------------------------

    --------------------------------

    --------------------------------

    (b) Name three priority queues.

    --------------------------------

    --------------------------------

    --------------------------------

    (c) Name a divide-and-conquer search algorithm, which only works on a sorted list.

    --------------------------------

    (d) Name an $O(n)$-time search algorithm, generally used only when $n$ is small.

    --------------------------------

12. Solve the recurrences, expressing answers using $\Theta$.

    (a) $F(n) = 2F(n/2) + n$

    (b) $F(n) = F(\sqrt{n}) + 1$

13. Find the asymptotic time complexity of each of these code fragments in terms of $n$, using $\Theta$ notation.

    (a)
    ```
    for(int i = 0; i < n; i++)
      for(int j = 1; j < i; j = 2*j);
    ```

    (b)
    ```
    for(int i = 1; i < n; i++)
      for(int j = i; j < n; j = 2*j);
    ```

    (c) `for(float x = n; x > 2.0; x = sqrt(x))` (`sqrt(x)` returns the square root of `x`.)

    (d)
    ```
    for(int i = 1; i < n; i = 2*i)
      for(int j = 2; j < i; j = j*j);
    ```
    (Hint: use substitution)

    (e)
    ```
    for(int i= 0; i < n; i++)
      for(int j = 0; j*j < n; j++)
    ```

    (f) `for(float i = n; i >= 1.0; i = log(i))`

    (g)
    ```
    for(int i = n; i > 1; i = i/2);
      for(int j = 1; j < i; j = 2*j);
    ```

14. Fill in the blanks.

    (a) The items in a priority queue represent ________________________ ________________________.

    (b) Name three kinds of search structures. ________________________________ ________________________________ ________________________________

15. Write the prefix expression equivalent to the infix epression $-a * b - (-c - d) \wedge e$
(Don't forget that $\wedge$ means exponentiation.)

16. Walk through the stack algorithm to change the infix expression $-a + b \wedge c \wedge -f$ to postfix. Show the stack at each step.

17. [20 points] Find an optimal prefix-free binary code for the following weighted alphabet. Show the Huffman tree.

| | |
|---|---|
| $a$ | 6 |
| $b$ | 4 |
| $c$ | 2 |
| $d$ | 5 |
| $e$ | 20 |
| $f$ | 1 |

18. Recall the Fibonacci numbers. Find a constant $K$ such that $F_n = \Theta(K^n)$.

19. In this problem, assume that it takes one time step to compute any addition or multiplication.

Consider the following recursive C++ function.

```
int f(int n)
 {
  if(n <= 0) return 0;
  else
   return f(n/6) + f(n/3) + f(n/2) + n;
 }
```

(a) Write a dynamic program which computes f(0) ... f(n) by dynamic progrmming, storing them in the following array.

```
int f[n+1];
```

What is the time complexity of your program?

(b) Write a recurrence for $f(n)$ and solve it, giving an asymptotic answer.

(c) Let t(n) be the time it takes for the above code to compute f(n). Write a recurrence for t(n) and solve it, giving an asymptotic answer.

(d) Write a dynamic program which computes f(0), f(1), ... f(n) by dynamic progrmming, storing them in the following array.

```
int f[n+1];
```

What is the time complexity of this dynamic program, in terms of n? Give an asymptotic answer.

(e) If you only need the value of f(n), instead of f(i) for all i in $0 \ldots$ n, you could use memoization. How many memos would you need to compute and store? Give an asymptotic answer, in terms of n. Hint: it's less than n.