COL100 Minor-1

KSHITIJ ALWADHI

TOTAL POINTS

39 / 60

QUESTION 1

Multiply using Add 20 pts

1.1 Simple Multiply 6 / 8

✓ **+ 6 pts Correct Algorithm**

+ 2 pts Correct Invariant/Induction Hypothesis

- 1 pts Minus Operator used

+ 0 pts Incorrect/Blank

1.2 Asymptotic Analysis 4 / 4

✓ **+ 2 pts Recurrence Relation**

✓ **+ 2 pts Correct Time Complexity and Space Complexity**

+ 1 pts Only Time complexity

+ 1 pts Only Space Complexity

+ 0 pts Incorrect / Not Attempted

What is n?

1.3 Fast Multiply 4 / 8

✓ **+ 4 pts Correct Algorithm**

+ 2 pts Correct Invariant/Induction Hypothesis

+ 1 pts Correct Recurrence Relation

+ 1 pts Correct Time Complexity

+ 0 pts Incorrect Solution/Blank

QUESTION 2

2 Chessboard Ways 10 / 10

+ 0 pts Used Formula

✓ **+ 10 pts Correct Algorithm with Base Case and Induction Step**

+ 3 pts Correct Base Case

+ 4 pts Correct Induction Hypothesis

+ 3 pts Correct Induction Step

+ 0 pts Unattempted/Wrong Answer

QUESTION 3

3 Divide 6 / 15

+ 0 pts No Induction Hypothesis or Base Case

+ 2 pts Partial Marks

✓ **+ 5 pts Base Case and Induction Hypothesis**

+ 5 pts Correct Induction Step

+ 2.5 pts Figured only one case

+ 3 pts Correct Recurrence

✓ **+ 2 pts Correct Time Complexity**

- 1 Point adjustment

You have confused n with x.
T(x) = T(x/2) + c or T(x) = T(x/2) + O(1) are correct.
OR
T(n) = T(n-1) + c or T(n) = T(n-1) + O(1) are correct.
The number of bits n decrements by one each call.
The number x itself gets halved each Iteration.
I am giving you marks this time with a penalty so that you keep the input in mind while writing the recurrence relation and Time complexity.

QUESTION 4

Iterative Integer Square Root 15 pts

4.1 Iterative Algorithm 6 / 12

✓ **+ 4 pts Figuring out c0 and c**

+ 4 pts i is the integer square root of n div 4c

+ 4 pts 2i, 2i+1 step

+ 0 pts Unattempted/Incorrect

+ 0 pts Recursive Approach

+ 2 Point adjustment

4.2 Asymptotic Analysis 3 / 3

✓ **+ 1.5 pts Correct number of steps**

✓ **+ 1.5 pts Recursive formulation given**

+ 0 pts Incorrect/Unattempted

COL 100	Minor 1	Feb 05, 2020

Name KSHITIJ ALWADHI *EntryNo* 2019EE30577 *Group* 30

Note (i) Write your answers in the blank parts of the question sheet including the back of the sheet. Do all rough work in separately provided paper. (ii) Write your answers neatly and precisely. You won't get a second chance to explain what you have written.

Problem 1: 20 marks

Suppose your favourite programming language can only perform an add (`add`: $\mathbb{N}\times\mathbb{N} \to \mathbb{N}$) operation, but not multiply.

1. Then, define multiplication (`mul`: $\mathbb{N} \times \mathbb{N} \to \mathbb{N}$) in terms of the `add`: $\mathbb{N} \times \mathbb{N} \to \mathbb{N}$ function (`add(a,b)` returns $a + b$).

```
fun mul(a,b) =
    let fun iter(a,b,ans,i) =
        if i=b then ans
        else iter(a, b, add(ans, a), add(i,1))
    in iter(a,b,0,0)
    end;
```

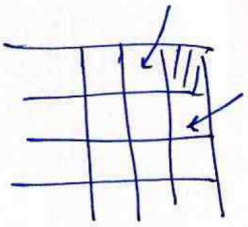
→ if a=0 then 0

2. Derive the time and space complexity of this algorithm in terms of number of invocations of the recursive function in the definition of `mult`?

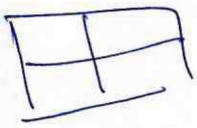
The iter function is called 'b' number of times as the counter i goes from 0 to b.

∴ The time complexity will be O(n)

and since this is an iterative definition, the space complexity will be O(1).



2×2



✓

3 × 4

↓

6 × 2 .

↓

12 × 1

3 × 7

↓

3 × 6 — 3

↓

6 × 3

↓

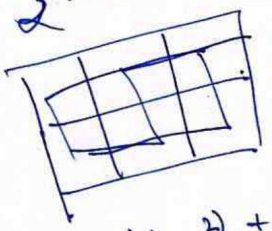
6 × 2 — 6

↓

(12 × 1)

12 + 9 = 21 .

2 × 3



ways(1,3) + ways(2,2)

↓ ↓

1 2

= (3)

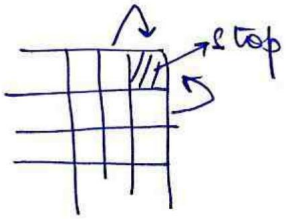
✓ .

3. Suppose we include, in addition to `add`, functions `dec` : $\mathbb{P} \to \mathbb{N}$, which decrements an integer by 1, `even` : $\mathbb{N} \to \mathbb{B}$, which tests whether or not a given number is even, `double` : $\mathbb{N} \to \mathbb{N}$, which doubles an integer, and `halve` : $\mathbb{N} \to \mathbb{N}$, which divides an (even) integer by 2. Can you construct a more efficient algorithm for `mult`.

```
fun mult (a,b) =
    let fun iter (a, b, aus) =
    if b = 1 then aus + a
    else if (even(b)) then iter (double(a), halve(b), aus)
    else iter (a, dec(b), aus + b) ;
    in iter (a, b, 0)
    end;
```

Problem 2: 10 marks

Write an `ML` program to count the number of ways for a rook to move from the southwest corner of a $p \times q$ chessboard to the northeast corner by moving one square at a time eastward or northward only. Note that rook is a chess piece that can move horizontally and vertically on a chess board. Give a recursive formulation and do not use a formula.



2 ways of reaching corner.

SML code:

```
fun ways (p,q) =
    if p = 1 orelse q = 1 then 1
    else ways (p-1, q) + ways (p, q-1) ;
```

(q, r) = div(1, 2)

(3, 2)

r_2 =

Problem 3: 15 marks

Consider the following algorithm for dividing an integer $x \geq 0$ by an integer $y > 0$ and determining $q \geq 0$ and $0 \leq r < y$ such $x = qy + r$.

```
fun divide(x,y) =
        if (x = 0) then (0,0)
        else  let   val (q,r) = divide(x div 2,y);
                    val q1 = 2*q;
                    val r1 = 2*r
              in
                    if (x mod 2 = 1) then
                          let val r2 = r1+1
                          in
                             if (r2 < y) then (q1,r2)
                             else (q1+1,r2-y)
                          end
                    else if (r1 < y) then (q1,r1)
                         else (q1+1,r1-y)
              end;
```

Prove that the algorithm is correct. Estimate the time complexity of the above algorithm as a function of n, where the number of digits/bits in x is n.

To prove: divide (x, y) gives (q, r) such that $x = qy + r$ and $0 \leq r \leq y$.

$x = 0$

Basis: divide $(0, y) = 0$ (by algo) which is true.

I.H: divide $(x, y) = (q, r)$ for some $0 < x < k$

I.S:

Time complexity $= O(\log_2 N)$.

(0, n, maxpow(~~(n div 4)~~))

$n = 53.$

(0, 53, 16)

↓

(, , 4)

$c_0 = 4^2 \quad k_0 = 2.$

$c = 16 \text{ div } 4^k.$

(2i)(2i)

Problem 4: 15 marks

In second lab assignment we had defined the *integer square root* of an integer n as the integer k such that $k^2 \leq n < (k+1)^2$, and computed it using the following inductive process:

- Compute the integer square root i of $m = n \; div \; 4$ recursively. We then have that $i^2 \leq m < (i+1)^2$
- Since m and i are integers we have that $(m+1) \leq (i+1)^2$. We thus have $(2i)^2 \leq 4m \leq n < 4m+4 \leq (2i+2)^2$. Hence we have that the integer square root of n is either $2i$ or $2i+1$.

1. Now write an iterative version corresponding to the above algorithm using the following invariant:

   INV : $c_0 = 4^{k_0}$ such that $n \; div \; 4^{k_0} > 0$ and $n \; div \; 4^{k_0+1} = 0$. After k iterations, $0 \leq k \leq k_0$, $c = c_0 \; div \; 4^k$ and $i^2 \leq n \; div \; (c * 4) < (i+1)^2$.

   You may use the following helper function:

```
fun maxpow(n) =
    if (n div 4 = 0) then 1
    else 4*maxpow(n div 4);
```

fun isqrt(n) = let

fun iter(i, n, c) =
  if (n div 4 = 0) then i
  else if ( ) then iter(, ,)
  ↳ condition is i
  c div 4.

in iter(0, n, maxpow(n))
end;

2. Derive the number of steps required by the algorithm.

Since we have $n \; div \; 4^{k_0} > 0$ and $n \; div \; 4^{k_0+1} = 0$

$T(N) = T(N/4) + 1$

$T(N/4) = T(N/4^2) + 1$

$\vdots$

$T(N) = T(0) + \log_4 N$

$T(N) = \log_4 N + 1$

The complexity is $O(\log_4 N)$