Lahore University Of Management Sciences
BSc (Honours) Programme
MS Computer Science Programme

Roll #: ________________________

Course Title: Computer Science Fundamentals
Course Code: CSI 11/CMPE 111
Instructor: Haroon Babri
Exam: Midterm
(mid-term/final/others)

Quarter: Winter
Academic Year: 2002-2003
Date: January 07, 2003
Time Allowed: 90 minutes
Total Marks: 100

DO NOT OPEN THIS EXAM UNTIL TOLD TO DO SO.

The instructions below must be followed strictly. Failure to do so can result in serious grade loss.

⇒ You may not
• talk to anyone once the exam begins.
• leave the examination room and then return.

⇒ Keep your eyes on your own paper.
⇒ Read all questions very carefully before answering them.

Specific instructions:


2. Calculator usage: Allowed

3. Write in pen/pencil: Pen Blue or Black

4. Any other instruction(s): ____________________________

__________________________
1. Prove the following relationship by induction:

\[ \sum_{i=1}^{n} (2i-1) = n^2. \]  \hspace{1cm} \text{(12)}

1. **Basis:** \( n = 1 \)
   
   \[ \text{L.H.S.} = 2(1)-1 = 1 \quad \text{R.H.S.} = 1^2 = 1 \] proved

2. **Inductive hypothesis:** the above eqn. is true.

3. **Proof:** for \( n+1 \),
   
   \[ \sum_{i=1}^{n+1} (2i-1) = \sum_{i=1}^{n} (2i-1) + [2(n+1)-1] \]
   
   (using Step 2)
   
   \[ = n^2 + 2n + 1 \]
   
   \[ = (n+1)^2 \]
   
   \[ \text{// what was required} \]

2. An application requires creating medical images from \( m \) data points collected by the imaging system. The running times of two algorithms A and B as a function of the data points are:

   \[ T(m) = 2m^4 \text{ grows faster} \]

   \[ T(m) = 28800m^2 \]

   For what values of \( m \) is algorithm A faster than algorithm B?

   Algo A is faster as long as
   
   \[ 2m^4 < 28800m^2 \]
   
   \[ m^2 < 14400 \]
   
   \[ m < 120 \]
   
   So for \( 0 < m < 120 \)
   
   A is faster
3. The following program fragment is intended to calculate the sum of odd and even integers up to n:

```c
scanf("%d", &n);
oddsum = 0;
evensum = 0;
for (j = 0; j <= n; j++)
   if (j % 2 != 0)
      oddsum = oddsum + j;
   else
      evensum = evensum + j;
printf("%d \n", oddsum);
printf("%d \n", evensum);
```

(a) Calculate the average running time T(n) of the above fragment

Assume that it takes one time unit each to do a scan, print, assignment, and testing a condition. Clearly show your work. You may mark the code above to show your work.

\[
T(n) = 4 + n + 2 + (n/2)(n/2) + \frac{n}{2} + \frac{n-1}{2} + 1.
\]

\[
T(n) = 4n + 10
\]

\[
T_{odd}(n) = 4 + n + 2 + (n/2)(n/2) + \frac{n+1}{2} + \frac{n+1}{2} + 1.
\]

\[
T_{odd}(n) = 4n + 10
\]

\[
T_{even} = 4n + 10
\]
4. The running time of a program is found to be
   \[ T(n) = n^2 + 6n + 2 \]

   (a) To prove that \( T(n) \) is \( O(n^2) \), is it possible to choose a pair of witnesses \((n_0, c)\) such that \( n_0 = 0 \)? Justify your answer.

\[
\begin{align*}
\text{To prove } T(n) &\text{ is } O(n^2) \\
90 + 6n + 2 &\leq c n^2 \quad \forall n \geq n_0 \\
\text{for } n_0 = 0 &\\
90 + 2 &\leq c \cdot 0 \\
\text{which cannot be satisfied for any } c > 0 \\
\text{Hence it is not possible to choose witnesses of type } (0, c)
\end{align*}
\]
5. The following program fragment computes the sum of integers in array X[0...p-1]:

```c
sumX = 0;
for(i=0; i<p; i++)
    sumX = sumX + X[i];
```

Find an appropriate loop invariant and use it to prove that the above fragment works as intended.

<table>
<thead>
<tr>
<th>i</th>
<th>sumX</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0+X[0]</td>
</tr>
<tr>
<td>2</td>
<td>0+X[0]+X[1]</td>
</tr>
<tr>
<td>k</td>
<td>[\sum_{i=0}^{k-1} X[i]]</td>
</tr>
</tbody>
</table>

\[k=0\]

1) **Base**: We reach the test the first time, with \(i=0\) and \(sumX = \sum_{i=0}^{1} X[i] = 0\) — Proved

2) **Assume**: \(S(k)\) is true

3) **To prove** \(S(k+1)\)

   a) For \(k > p\)

   For this case we break at \(k+1\) earlier and hence never reach the test, so \(S(k+1)\) is true.

   b) For \(k < p\)

   \[
   S(k) = \sum_{i=0}^{k-1} X[i], \quad k \leq k
   \]

   \[
   S(k+1) = S(k) + X[k], \quad k \rightarrow k+1
   \]

   \[
   = \sum_{i=0}^{k-1} X[i] + X[k]
   \]

   \[
   = \sum_{i=0}^{k} X[i]
   \]

(25)
6. (a) Write a recursive function `findseq` that returns TRUE if two integers x and y occur in sequence in a list L and FALSE otherwise. For example if L = (2, 10, 14, 5, 1, 3) then the function call `findseq(1,2,L)` should return FALSE whereas `findseq(3,1,L)` should return TRUE. Assume that the list L is implemented as a linked list.

(b) Write a function `multiseq` that returns an integer value indicating the number of times two integers x and y occur in sequence in a list L, assuming that L is implemented as an integer array. For example if L = (1,2,4,6,3,2,4) then the function call `multiseq(2,4,...)` will return 2. Assume that a list can have a maximum of 1000 elements.

```c
BOOL Findseq(int x, int y, LIST *pl)
{
    if
        if ((*pl) -> next == NULL)
            return FALSE;
        if ((*pl) -> element == x) {
            (*pl) = (*pl) -> next;
            if ((*pl) -> element == y)
                return TRUE;
        }
    else
        Findseq(x, y, &(*pl) -> next);
}
```
int multiseq (int x, int y, L *pl)
{
    int i = 0, count = 0;  // & pL -> A[i+1] == NULL
    if (pl -> A[i] == NULL)
        return count;
    else
    {
        for (i = 0; i < pl -> length; i++)
            if (pl -> A[i] == x)
                if (pl -> A[i+1] == y)
                    count++;
        return count;
    }
}