22**C** : 031 **Algorithms**
Midterm

This is a closed book exam. You have an hour and fifteen minutes.

1. Give an asymptotically tight bound on the worst case running time of the following algorithm as a function of n, the number of elements in input array A and output array C. (Express runing time as $\Theta(f(n))$ for some appropriate f.) (2 points)

```
For i from 1 to n do
C[i] := 0
endfor

For i from 1 to n do
 For j from i to n do
   C[i] := C[i] + A[j]
 endfor
endfor
Return C
```

2. Give an asymptotic upper bound on the worst case running time of the following algorithm as a function of n, the number of elements in input array A and output array C. (Express running time as $O(f(n))$ for some appropriate f.) Pick as good an f as you can. (3 points)

```
For i from 1 to n do
C[i] := 0
endfor

For i from 1 to n do
 j:= i
 While j is less than or equal to n do
   C[i] := C[i] + A[j]
   j:= 2 * j
 endwhile
endfor
Return C
```

3. In each of the following cases, say whether $f(n)$ is $O(g(n))$ and whether $f(n)$ is $\Omega(g(n))$. For example, if $f(n) = n^2$ and $g(n) = n^3$, then $f(n)$ is $O(g(n))$ and $f(n)$ is not $\Omega(g(n))$. (2 points)

   (a) $f(n) = n \log n$, $g(n) = n^2$.

(b) $f(n) = 100n^2 + 300n$, $g(n) = n^2$.

(c) $f(n) = \frac{n^2}{3} - 200n + 120000$, $g(n) = n^2$.

(d) $f(n) = 1.17^n$ and $g(n) = 100n^2$.

4. Consider the stable matching problem involving the three men m_1, m_2, m_3 and the three women w_1, w_2, w_3 with the following preferences:

   - $m_1 : w_1 > w_3 > w_2$
   - $m_2 : w_1 > w_2 > w_3$
   - $m_3 : w_3 > w_1 > w_2$
   - $w_1 : m_2 > m_3 > m_1$
   - $w_2 : m_1 > m_2 > m_3$
   - $w_3 : m_1 > m_3 > m_2$

   Is the perfect matching that matches m_1 to w_1, m_2 to w_2, and m_3 to w_3 stable? If not, identify an instability, and describe a stable matching. (2 points)

5. Consider the two recursive algorithms we discussed for multiplying two n-polynomials when n is an integer power of 2. (3 points)

   (a) When we call the $\Theta(n^2)$ recursive algorithm for multiplying two n-polynomials, what is the total number of base case instances that are solved? Recall that in a base case instance we multiply two 1-polynomials. You can give the answer as an exact expression in terms of n, or in the form $\Theta(f(n))$ for some appropriate f.

   (b) When we call the $O(n^{\log_2 3})$ recursive algorithm for multiplying two n-polynomials, what is the total number of base case instances that are solved?

6. Consider the $O(n \log^2 n)$ algorithm we discussed in class (or the $O(n \log n)$ algorithm in the textbook) for finding the closest pair in a given set P of n points in the plane. We partitioned P into two sets P_1 and P_2 of roughly equal size so that points in P_1 have x-coordinates that are less than or equal to the x-coordinate of each point in P_2. We recursively computed the closest pair within P_1 and the closest pair within P_2 and then followed these up by considering pairs in $P_1 \times P_2$.

   Suppose the algorithm design is changed so that P_1 and P_2 are obtained by partitioning according to y-coordinates rather than x-coordinates. That is, we sort P by y-coordinates, let P_1 be the first half of P in this sorted order and P_2 be the second half. Describe how the rest of the algorithm is to be modified – give the pseudocode. (3 points)