**Problem Set 1** **Due: Feb 2** **cs2me3 Winter 2005**

## Instructions

1. You are encouraged to work in groups of two. If you cannot find a partner, you can work alone.
2. Please submit one copy of the assignment; if you are working with a partner, both names should appear on the assignment.
3. For **Part A** of the assignment, you must submit an electronic copy of your C program via WebCT, and a hard copy of your program should be included when you hand in the assignment.

## Part A

Write a C program which implements the Huffman code technique for compressing data. (Section 16.3 in the textbook.)

Your program should work as follows: given an ASCII text file as input, `file.txt`, it should output a compressed binary file `file.bin`. Your program has to work in both directions (given `file.bin`, it produces `file.txt`).

The direction will be specified with a switch: `-c` for compressing, and `-x` for extracting.

Thus, the result of `huffman -c file.txt` has to be `file.bin`, and the result of `huffman -x file.bin` has to be `file.txt`.

Give a description of your implementation, and specify the limitations of your program (for example, what happens if you do `huffman -x file.txt`?).

## Part B

1. This question is an induction warm up. Suppose that we have a (Swiss) chocolate bar consisting of a number of squares arranged in a rectangular pattern. The task is to split the bar into small squares (always breaking along the lines between the squares) with a minimum number of breaks. How many will it take? Make an educated guess, and prove it by induction.

   **Solution:** Assertion: a chocolate bar with $n$ squares requires $(n-1)$ breaks. We now prove this assertion by induction.

   Basis Case: Only one square needs 0 breaks, so we are OK.

   Induction Step: Suppose that we have $m+1$ squares. We break the bar into two smaller pieces of $a$ and $b$ squares each, $a+b=m+1$.

   By induction hypothesis, the "$a$" piece requires $a-1$ breaks, and the "$b$" piece requires $b-1$ breaks, so together the number of breaks is $(a-1)+(b-1)+1 = a+b-1 = m$, and we are done.

2. In this question you are going to complete the proof of correctness of the job scheduling algorithm. Your task is to give clear and concise answers to the five "why's" in the segment of the proof given below.

   **case (II)** job $i$ is scheduled at time $t_0$, so $S'(t_0) = i$ (whereas $S(t_0) = 0$) and $t_0$ is the latest possible time for job $i$ in the schedule $S$.

   **subcase (a)** job $i$ is scheduled in $S_{\text{opt}}$ at time $t_1$:
   If $t_1 = t_0$ then, as in case (I), just let $S'_{\text{opt}} = S_{\text{opt}}$.
   If $t_1 < t_0$ then let $S'_{\text{opt}}$ be $S_{\text{opt}}$ except that we interchange $t_0$ and $t_1$, that is: $S'_{\text{opt}}(t_0) = S_{\text{opt}}(t_1) = i$ and $S'_{\text{opt}}(t_1) = S_{\text{opt}}(t_0)$. Then $S'_{\text{opt}}$ is feasible (**why1?**), it extends $S'$ (**why2?**), and $P(S'_{\text{opt}}) = P(S_{\text{opt}})$ (**why3?**).
   The case $t_1 > t_0$ is not possible (**why4?**).

   **subcase (b)** job $i$ is not scheduled in $S_{\text{opt}}$. Then we simply define $S'_{\text{opt}}$ to be the same as $S_{\text{opt}}$, except $S'_{\text{opt}}(t_0) = i$. Since $S_{\text{opt}}$ is feasible, so is $S'_{\text{opt}}$, and since $S'_{\text{opt}}$ extends $S'$, we only have to show that $P(S'_{\text{opt}}) = P(S_{\text{opt}})$. This follows from the following claim:
   **Claim:** Let $S_{\text{opt}}(t_0) = j$. Then $g_j \leq g_i$. (**why5?**)

   We answer all the whys:

   **why1** To show that $S'_{\text{opt}}$ is feasible, we show have to show that no job is scheduled twice, and no job is scheduled after its deadline. The first is easy, because $S_{\text{opt}}$ was feasible. For the second we argue like this: the job that was at time $t_0$ is now moved to $t_1 < t_0$, so certainly if $t_0$ was before its deadline, so is $t_1$. The job that was at time $t_1$ (job $i$) has now been moved forward to time $t_0$, but we are working under the assumption that job $i$ was scheduled (at this point) in slot $t_0$, so $t_0 \leq d_i$, and we are done.

**why2** $S'_{\text{opt}}$ extends $S'$ because $S_{\text{opt}}$ extended $S$, and the only difference is positions $t_1$ and $t_0$. They coincide in position $t_0$ (both have $i$), so we only have to examine position $t_1$. But $S(t_1) = 0$ since $S_{\text{opt}}(t_1) = i$, and $S$ does not schedule job $i$ at all. Since the only difference between $S$ and $S'$ is in position $t_0$, it follows that $S'(t_1) = 0$, so it does not matter what $S'_{\text{opt}}(t_1)$ is, it will extend $S'$.

**why3** They schedule the same set of jobs, so they must have the same profit.

**why4** Suppose $t_1 > t_0$. Since $S_{\text{opt}}$ extends $S$, it follows that $S(t_1) = 0$. Since $S_{\text{opt}}(t_1) = i$, it follows that $t_1 \leq d_i$. But then, the algorithm would have scheduled $i$ in $t_1$, not in $t_0$.

**why5** We prove the claim by contradiction: assume that $g_j > g_i$ (note that in this case $j \neq 0$). Then job $j$ was considered before job $i$. Since job $i$ was scheduled at time $t_0$, job $j$ must have been scheduled at time $t_2 \neq t_0$ (we know that job $j$ was scheduled in $S$ since $S(t_0) = 0$, and $t_0 \leq d_j$, so there was a slot for job $j$, and therefore it was scheduled). But $S_{\text{opt}}$ extends $S$, and $S(t_2) = j \neq S_{\text{opt}}(t_2)$ — contradiction.