Solutions to Practice Problems for the Final Exam

Disclaimer: This just reflects the material since the second midterm.

Solution 2:

(i) boolean isValid(int k, int x): A block at level \( k \) starts at address \( x \) if \( x \) is a multiple of \( 2^k \), or equivalently its \( k \) lowest-order bits are all zero. That is, \((\text{bitMask}(k) \& x) == 0\).

(ii) int sibling(int k, int x): As given in class, this comes about by complementing the \( k \)th order bit of \( x \) (where the least significant bit is bit 0), that is, \((1<<k)^x\). For example:

\[
\text{sibling}(2, 12) = \text{sibling}(2, 001100_2) = 000100_2 ^ 001100_2 = 001000_2 = 8.
\]

(iii) int parent(int k, int x): To obtain the parent, we zero out the \( k \)-order bit, that is, \({\sim}(1<<k) \& x\). For example:

\[
\text{parent}(2, 12) = \text{parent}(2, 001100_2) = 000100_2 \& 001100_2 = 001000_2 = 8.
\]

(iv) int left(int k, int x): The left child’s starting address is the same as the parent’s starting address, so this is just \( x \).

(v) int right(int k, int x): The right child’s starting address is the sibling of the left child’s starting address at level \( k - 1 \), that is \((1<<(k-1))^x\). For example:

\[
\text{right}(2, 12) = \text{sibling}(1, 12) = 000010_2 \& 001100_2 = 001110_2 = 14.
\]

Solution 3: We maintain two pointers \( p \) (source) and \( q \) destination.

\[
\begin{align*}
(\text{void}*) \ & \text{compact}(\text{void}* \text{ start, void}* \text{ end}) \ {\text{\{}} \ & \text{compact memory from start to end-1} \\
\text{void} * \ & p = \text{start}; \ & \text{p points to source block} \\
\text{void} * \ & q = \text{start}; \ & \text{q points to destination block} \\
\text{while}(p < \text{end}) \ {\text{\{}} \ & \text{allocated block?} \\
\text{if}(\text{p.inUse}) \ {\text{\{}} \ & \text{copy to destination} \\
\quad \text{memcpy}(q, p, p.\text{size}); \ & \text{previous block is in-use} \\
\quad q.\text{prevInUse} = 1; \ & \text{increment destination pointer} \\
\quad q += p.\text{size}; \ & \text{(no need to set q.size or q.inUse, since they are copied from p)} \\
\text{\}}} \ & \text{advance to the next block} \\
\text{\}}} \ & \text{everything copied - now q points to the remaining available block} \\
q.\text{inUse} = 0; \ & \text{this block is available} \\
q.\text{prevInUse} = 1; \ & \text{previous block is in-use} \\
\text{int blockSize} = p - q; \ & \text{size of this final block} \\
q.\text{size} = \text{blockSize}; \ & \text{set q.size} \\
*(q + q.\text{size }-1) = \text{blockSize}; \ & \text{... and q.size2} \\
\text{return q; \ & \text{return pointer to this block} 
\end{align*}
\]