1. Consider the partially specified B+-tree in Figure 1.

(a) See Figure 2.
(b) See Figure 3.
(c) See Figure 4.

2. See Figure 5.

3. (1) (a) Match, Sailors.sid < 50000.
(a) Match, Sailors.sid = 50000.

(2) (a) No Match.
(b) Match, Sailors.sid = 50000.

(3) (a) Match, Sailors.sid < 50000 ∧ Sailors.age = 21.
(b) Match, Sailors.sid=50000 ∧ Sailors.age>21.
(c) Match, Sailors.sid = 50000.
(d) No match.
(4) This question can be understood in two ways:

(i) The textbook has a typo and there is only a hash-index on 
\((\text{Sailors}.\text{id}, \text{Sailors}.\text{age})\):

(a) Match, \text{Sailors}.\text{sid} = 50000 \land \text{Sailors}.\text{age} = 21

(b) No match.
Figure 4: Solution to problem 1(c)

Figure 5: Possible solution to problem 2. Insertion of 128* causes recursive split of order 4.

(c) No match.
(d) No match.
(ii) There is a hash and a B+-tree index on \( (\text{Sailors.id, Sailors.age}) \):
(a) Match, Sailors.sid = 50000 \( \land \) Sailors.age = 21 (Hash and B+-tree)
(b) Match, Sailors.sid = 50000 \( \land \) Sailors.age > 21 (B+-tree)
(c) Match. Sailors.sid = 50000 (B+-tree)
(d) No match.

4. Consider the following SQL query

\[
\text{SELECT ROADID} \\
\text{FROM ROADS R, ZONES Z1, ZONES Z2} \\
\text{WHERE R.SRCZONE = Z1.ZONEID AND R.ENDZONE = Z2.ZONEID AND} \\
\text{Z1.TYPE = 'R' AND Z2.TYPE = 'C' AND R.DIST < 10}
\]

(a) Please note the linebreak after ... \((ZONES)\times!\)

\[
\Pi_{\text{ROADID}}(\sigma_{\text{SRCZONE}=\text{ZONEID}_1\land\text{ENDZONE}=\text{ZONEID}_2} \quad \text{ROADS}\times\rho_{X_1}(\text{ZONES})\times \\
\rho_{X_2}(\text{ZONES}))
\]

(b) Our assumption is that the ZONE table is much smaller than the ROADS table. The query tree of the naive evaluation is depicted in Figure 6. In Figure 7 one can see the tree after the selections have been pushed down, and in Figure 8 after the Cartesian products have been rewritten as joins.
Figure 6: Query tree of naive evaluation.
Figure 7: Query tree after the selections have been pushed down.
Figure 8: Query tree after making Cartesian products into joins.