Student Name: _____
Student ID: _____

1. Indexing is:
a) Random access to an array.
b) The process of associating a key with the location of a corresponding data record.
c) Using a hash table.

2. The following pseudo code

```
if search key == one of the keys in the current node
   return corresponding record (using pointer associated with key)
else
   search in child nodes
```

does NOT describe a part of the search algorithm for the
a) 2-3 tree
b) B tree
c) B+ tree
d) All of the above

3. Consider the code fragment below.

```
bool find (nodeptr, key, recordptr){
       if (nodeptr is a leaf){
              if (key==left key) recordptr=left keyptr;
              else if (key==right key) recordptr=right keyptr;
              else return false;
              return true;}
       if (key < left key) find (leftchildptr, key, recordptr);
       else if (key < right key) find (centerchildptr, key, recordptr);
       else if (key >= right key) find (rightchildptr, key, recordptr);}
```

It is used to find a record in a:
a) 2-3 tree
b) B Tree
c) B+ Tree
d) All of the above
e) None of the above

4. When inserting a key of a record in a B+ tree, if the node that should contain this key (along with the pointer to the record) is full, this node should be split into two nodes (add a new node and divide the key-pointer pairs evenly among the two nodes) and
a) Promote the middle key in the old node
b) Promote the middle key in the new node
c) Promote a copy of the middle key in the new node
d) Promote the smallest key in the new node
e) Promote a copy of the smallest key in the new node

5. The primary key is:
a) A unique identifier for a record.
b) The main search key used by users of the database.
c) The first key in the index.

6. Suppose `root` is a pointer to a B+ tree node, `A` is the array of key/pointer pairs for this node, `n` is the number of elements, `k` is the search key, and `e` is the pointer to the record we are searching for. Each element of `A` consists of the pair `ky` and `ptr`. If the functions:

`gle(PAIR A[],int n,KEY k)` returns the greatest value in `A` less than or equal to `k`

`sge(PAIR A[],int n,KEY k)` returns the smallest value in A greater than or equal to `k`

Then, the following code fragment searches for the record whose key is `k` in an index file that uses a B+ tree, whose root node is pointed to by `root`

a)
```
bool find(BPNode *root, Key k, Elem *e){
     cur=sge(root->A, root->n,k);
       if (root->isleaf())
          if (root->A[cur].ky==k){ e= root->A[cur].ptr; return true;}
          else return false;
       else find(e= root->A[cur].ptr, k, e);}
```

b)
```
bool find(BPNode *root, Key k, Elem *e){
     cur=sge(root->A, root->n,k);
       if (root->isleaf())
          if (root->A[cur].ky==k){ e= root->A[cur].ptr; return true;}
          else find(e= root->A[cur].ptr, k, e);}
```

c)
```
bool find(BPNode *root, Key k, Elem *e){
     cur=gle(root->A, root->n,k);
       if (root->isleaf())
          if (root->A[cur].ky==k){ e= root->A[cur].ptr; return true;}
          else return false;
       else find(e= root->A[cur].ptr, k, e);}
```

d)
```
bool find(BPNode *root, Key k, Elem *e){
     cur=gle(root->A, root->n,k);
       if (root->isleaf())
          if (root->A[cur].ky==k){ e= root->A[cur].ptr; return true;}
          else find(e= root->A[cur].ptr, k, e);}
```

7. Assume that a computer system has disk clusters (blocks) of 256 bytes (¼KB) and the language used to develop software uses 4 bytes for integers, 4 bytes for pointers, 1 byte for characters, and 8 bytes for doubles, and 8 bytes for longs. Assume we have the following records (the name filed is 32 characters, ID is a long, and Pay is a double):

Name	Flor	Saab	Jaber	Safa	Flor	Kadi	Harb	Harb	Safa	Harb	Kadi	Safa
ID	2000	2010	2011	2100	2110	2017	2019	2155	2066	2022	2013	2090
Pay	56.2	57.8	56.0	55.0	60.1	58.4	57.6	56.16	59.4	63.1	62.9	45.6

If we use a two-dimensional linear index, using the Name as secondary key and ID as primary key, then the size of the index file in bytes will be:

a) 256 bytes
b) 512 bytes
c) 528 bytes
d) 768 bytes
e) 1024 bytes

8. B+ trees are efficient for
a) Less than queuries
b) Greater than queuries
c) Range queuries
d) All of the above
e) None of the above

9. Tree indexing methods are meant to overcome what deficiency in linear indexing?
a) Difficulty to handle range queries.
b) Difficulty in handling updates.
c) Difficulty to handle large data sets.

10. The most important advantage of a 2-3 tree over a BST is that:
a) The 2-3 tree has fewer nodes.
b) The 2-3 tree has a higher branching factor.
c) The 2-3 tree is height balanced.

11. The B-tree:
a) grows vertically up
b) grows vertically down
c) grows horizontally

12. The primary difference between a B-tree and a B+-tree is:
a) the adjacent nodes are linked
b) The B+-tree store records only at the leaf nodes.
c) The B+-tree is hight balanced.
d) The B+-tree is smaller.

1. b
2. c
3. e
4. e
5. a
6. c
7. b
8. d
9. b
10. c
11. a
12. b