```
function [category_index predicted_z final_delta] =
predict_best_fit_tree(anchor_tree, delta_tree, new_data_item, predict_z_toggle,
search_toggle)

%search_toggle must be -1,0, or 1;
%this determines whether the algorithm terminates upon the first match, best match
at first depth, or best match in the entire tree, respectively

[width depth] = size(anchor_tree);
new_data_vector = new_data_item{1};

%this is to ensure we ignore any labels in the data
new_data_vector = new_data_vector(1:3);

%We initialize the return values to indicate a failure to categorize
min_difference = Inf;
final_delta = Inf;

i = 0;
break_loop_depth = 0;
break_loop_width = 0;
match = 0;

while(i < depth && break_loop_depth == 0)

  j = 0;

  while(j < width && break_loop_width == 0)

    anchor_vector = anchor_tree{width - j, depth - i};
    delta = delta_tree{width - j, depth - i};

    if(isempty(anchor_vector) == 0)

      if(predict_z_toggle == 1)

        new_data_vector(3) = anchor_vector(3);

      endif

      temp_anchor = anchor_vector(1:3);

      difference = norm(new_data_vector - temp_anchor);

      %Either we have a match, or we've reached the top of the tree
      if(difference <= delta || anchor_vector(3) == Inf)

        %if true, then we have a match
```

```
        if(difference < min_difference)

          match = 1;
          predicted_z = new_data_vector(3);
          category_index = [(width - j) (depth - i)];
          final_delta = delta;
          min_difference = difference;

          %if true, then we terminate upon first match
          if(search_toggle == -1)

            break_loop_depth = 1;
            break_loop_width = 1;

          %if true, then we terminate at the end of the current depth
          elseif(search_toggle == 0)

           break_loop_depth = 1;

          %otherwise, we search the entire tree

          endif

        %if true, then we've reached the top of the tree with no match
        elseif(anchor_vector(3) == Inf && match == 0)

          predicted_z = Inf;
          category_index = [(width - j) (depth - i)];

        endif

      endif

    endif

    j = j + 1;

  endwhile

  i = i + 1;

endwhile

endfunction
```