```cpp
#include <pcl/point_cloud.h>
#include <pcl/octree/octree.h>

#include <iostream>
#include <vector>
#include <ctime>

int
main (int argc, char** argv)
{
  srand ((unsigned int) time (NULL));

  // Octree resolution - side length of octree voxels
  float resolution = 32.0f;

  // Instantiate octree-based point cloud change detection class
  pcl::octree::OctreePointCloudChangeDetector<pcl::PointXYZ> octree
(resolution);

  pcl::PointCloud<pcl::PointXYZ>::Ptr cloudA (new
pcl::PointCloud<pcl::PointXYZ> );

  // Generate pointcloud data for cloudA
  cloudA->width = 128;
  cloudA->height = 1;
  cloudA->points.resize (cloudA->width * cloudA->height);

  for (size_t i = 0; i < cloudA->points.size (); ++i)
  {
    cloudA->points[i].x = 64.0f * rand () / (RAND_MAX + 1.0f);
    cloudA->points[i].y = 64.0f * rand () / (RAND_MAX + 1.0f);
    cloudA->points[i].z = 64.0f * rand () / (RAND_MAX + 1.0f);
  }

  // Add points from cloudA to octree
  octree.setInputCloud (cloudA);
  octree.addPointsFromInputCloud ();

  // Switch octree buffers: This resets octree but keeps previous tree
structure in memory.
  octree.switchBuffers ();

  pcl::PointCloud<pcl::PointXYZ>::Ptr cloudB (new
pcl::PointCloud<pcl::PointXYZ> );

  // Generate pointcloud data for cloudB
  cloudB->width = 128;
  cloudB->height = 1;
  cloudB->points.resize (cloudB->width * cloudB->height);

  for (size_t i = 0; i < cloudB->points.size (); ++i)
  {
    cloudB->points[i].x = 64.0f * rand () / (RAND_MAX + 1.0f);
    cloudB->points[i].y = 64.0f * rand () / (RAND_MAX + 1.0f);
```

```
    cloudB->points[i].z = 64.0f * rand () / (RAND_MAX + 1.0f);
  }

  // Add points from cloudB to octree
  octree.setInputCloud (cloudB);
  octree.addPointsFromInputCloud ();

  std::vector<int> newPointIdxVector;

  // Get vector of point indices from octree voxels which did not exist
in previous buffer
  octree.getPointIndicesFromNewVoxels (newPointIdxVector);

  // Output points
  std::cout << "Output from getPointIndicesFromNewVoxels:" << std::endl;
  for (size_t i = 0; i < newPointIdxVector.size (); ++i)
    std::cout << i << "# Index:" << newPointIdxVector[i]
              << "  Point:" << cloudB->points[newPointIdxVector[i]].x <<
" "
              << cloudB->points[newPointIdxVector[i]].y << " "
              << cloudB->points[newPointIdxVector[i]].z << std::endl;

}
```