```cpp
#include <iostream>
#include <memory>
#include <cstdlib>
#include <vector>
#include <iterator>
#include <algorithm>

using namespace std;

struct Point {
  int x;
  int y;
  int z;

  Point() : x(1), y(2), z(3) {}
  Point(int x, int y, int z) : x(x), y(y), z(z) {}
  ~Point() {
    cout << "Destructor called for point: " << *this << "\n";
  }

  friend ostream& operator<<(ostream& os, const Point& p) {
    return (os << "{" << p.x << ", " << p.y << ", " << p.z << "}");
  }

};

int main() {
  vector<Point> points = {{2, 2, 4}, {-1, 6, 8}, {10, 12, 1}, {3, 2, 1}};
  void* mem = nullptr;
  std::size_t size = sizeof(Point) * points.size();

  mem = std::malloc(size);
  Point* tmp = static_cast<Point*>(mem);

  for_each(tmp, tmp + points.size(), [](Point& i) {
      cout << i << " @ " << &i << "\n";
    });


  uninitialized_default_construct(tmp, tmp + points.size());

  for_each(tmp, tmp + points.size(), [](Point& i) {
      cout << i << " @ " << &i << "\n";
    });

  uninitialized_copy_n(begin(points), points.size(), static_cast<Point*>(mem));

  for_each(tmp, tmp + points.size(), [](Point& i) {
      cout << i << " @ " << &i << "\n";
    });

  std::destroy(tmp, tmp + points.size());
  std::free(mem);

  cout << "End of program\n";

  return 0;
}
```