Paul Mallard & Pauline Dos Santos E4FI group 2

Question 5 : Given a **myMesh** M, and a **myVertex** v, give pseudo-code to delete v from M, and then re-triangulate the “hole” that results from removing v.

```
void deleteV(myMesh* M, myVertex* v){
   std::vector<myVertex*> neighbors;
   myHalfedge* start = v->halfedge;
   myHalfedge* e = start;

   do{
      if (e == NULL || e->twin == NULL){
         break;
      }
      else{
        myVertex* neighbor = e->twin->source;
        neighbors.push_back(neighbor);
        e = e->twin->next;
      }
   } while (e != start);

   //delet faces
   e = start;
   do {
      myFace* f= e->adjacent_face;

      for (int i = 0; i < M->faces.size(); ++i){
         if (M->faces[i] == f){
            for (int j = i; j < M->faces.size() - 1; ++j){
               M->faces[j] = M->faces[j + 1];
            }
            M->faces.pop_back();
            break;
         }
      }
      delete f;

      e = e->twin->next;
   } while (e != start);

   //delete halfedges and twins
   e = start;
   do {
      myHalfedge* twin = e->twin;

      // delete e from M->halfedges
      for (int i = 0; i < M->halfedges .size(); ++i){
         if (M->halfedges [i] == e){
```

```
            for (int j = i; j < M->halfedges .size() - 1; ++j){
                M->halfedges [j] = M->halfedges [j + 1];
            }
            M->halfedges .pop_back();
            break;
        }
    }
    delete e;

    //delete twins
    if (twin != NULL) {
        for (int i = 0; i < M->halfedges .size(); ++i) {
            if (M->halfedges [i] == twin) {
                for (int j = i; j < M->halfedges .size() - 1; ++j) {
                    M->halfedges [j] = M->halfedges [j + 1];
                }
                M->halfedges .pop_back();
                break;
            }
        }
        delete twin;
    }

    e = twin->next;
} while (e != start);

//delete v of vector M->vertices
for (int i = 0; i < M->vertices.size(); ++i) {
    if (M->vertices[i] == v) {
        for (int j = i; j < M->vertices.size() - 1; ++j) {
            M->vertices[j] = M->vertices[j + 1];
        }
        M->vertices.pop_back();
        break;
    }
}
delete v;

//retriangulate
int n = neighbors.size();
for (int i = 1; i < n - 1; ++i) {
    myVertex* v0 = neighbors[0];
    myVertex* v1 = neighbors[i];
    myVertex* v2 = neighbors[i + 1];

    //create halfedges
    myHalfedge* halfedges0 = new myHalfedge();
    myHalfedge* halfedges1 = new myHalfedge();
```

```
        myHalfedge* halfedges2 = new myHalfedge();

        halfedges0->source = v0;
        halfedges1->source = v1;
        halfedges2->source = v2;

        halfedges0->next = halfedges1;
        halfedges1->next = halfedges2;
        halfedges2->next = halfedges0;

        //create new face
        myFace* newFace = new myFace();
        newFace->halfedge =halfedges0;

        halfedges0->adjacent_face = newFace;
        halfedges1->adjacent_face = newFace;
        halfedges2->adjacent_face = newFace;

        M->faces.push_back(newFace);
        M->halfedges.push_back(halfedges0);
        M->halfedges.push_back(halfedges1);
        M->halfedges.push_back(halfedges2);
    }
}
```