These are example long answer questions. The final exam will have three long answer questions, each worth 20 points.

9. Given the following declaration for a `Stack` class, where `table` is an array of size `tablesize`, write the definitions of the default constructor, and the `push()` and `pop()` methods for the class. Don't worry about any `#includes`. To explain further, `tablesize` should be initially set to 0 by the default constructor, but this should be modified by `push()`, which will allocate new space as needed to grow the size of `table`.

```
class Stack{
private:
  int top;
  int tablesize;
  float *table;
public:
  Stack();
  Stack(const Stack &s);
  void clear();
  bool empty() const;
  void push(float number);
  float pop();
  Stack& operator=(const Stack &s);
};

Stack::Stack(): top(0), tablesize(0), table(NULL){
}

Stack::push(float number){
  if(top == tablesize){
    // the 10 below is arbitrary, anything > 0 will work
    float *resizedtable = new float[tablesize + 10];
    for(int i = 0; i < tablesize; i++)
      resizedtable[i] = table[i];
    delete table;
    table = resizedtable;
  }
  table[top] = number;
  top++;
}

float Stack::pop(){
  if(empty()){
    cerr << "stack underflow" << endl;
    exit(1);
  }
  top--;
  return table[top];
}
```

10. You are building a maze, which is organized as a 2D grid layout of a set of rooms. Each room can be either a wall, a corridor, a boss lair, an infirmary, or a treasure trove. Each room will be displayed with a different texture map, and each has a different effect on the player's movement. However, each room shares certain information, such as its location in the maze, and whether it has been entered before or not. Design a base class and subclasses for wall, boss, and treasure trove rooms.

```
// Most Room methods are pure virtual methods.
// You would never allocate a room, just subclasses
class Room{
protected:
  int textureid;
  bool visited;
  int row, col;
  bool doors[4];
public:
  Room() = 0;
  void Visit();
  bool WasVisited() const;
  virtual void SetTexture() = 0;
  virtual int HealthEffect() = 0;
  virtual int Reward() const = 0;
  virtual void Draw() const = 0;
}

class Wall: public Room{
public:
  Wall();
  void SetTexture();
  int HealthEffect() const; // would always return 0
  int Reward() const;       // would always return 0
  void Draw() const;
}
class Boss: public Room{
public:
  Boss();
  void SetTexture();
  int HealthEffect() const; // return -1 if not visited
  int Reward() const;       // would always return 0
  void Draw() const;
}
class Treasure: public Room{
public:
  Treasure();
  void SetTexture();
  int HealthEffect() const; // would always return 0
  int Reward() const;       // if not visited return treasure value
  void Draw() const;
}
```