2D: SAT Solver

Gwee Yong Ta 1004114
Filbert Cia 1004415
Tan Joon Kang 1004150
Ho Xin Yi Felice 1004605
Yu Nicole Frances Cabansay 1004574

Approach Taken to Obtain a Solution After Parsing the CNF File

We have used the DPLL Algorithm to find the solution to the Boolean Satisfiability problem.

```
def Environment solve(ImList<Clause> clauses, Environment env){
       If list of clauses is empty: return environment
       If any clause in the list of clauses is empty: return null

       Else:
              For element in list of clauses:
                     Find the smallest clause.
                            If the smallest clause has only 1 literal and it is
                            false,the formula will be unsatisfiable.
                            Else:
                                   We will pick an arbitrary literal from the smallest
                                   Clause, set it to true and solve recursively.
                                   If it fails, we then set it to false and solve it
                                   recursively.
```

What Have Done to Reduce Computational Time

There were certain instances where we used multiple for loops to iterate through the clauses. However, we realised that it is redundant and it takes longer to run. Hence, we put it all together in one for loop.

We also used Unit Propagation. In order to speed up the running time, we simplified the formula by finding the smallest clause and eliminating variables that prove to be unsatisfiable.

```
for(Clause currClause: clauses){
   if(currClause.size() < smallestClause.size()){
       smallestClause = currClause;
   }
   if(smallestClause.isUnit()){
       break;
   }
}
```

Result, Running Time, Computer Specifications

Not satisfiable
Time: 1412.5751ms
Acer Predator Helios, Intel core i7-8750H @2.20Ghz