***Question 1 : Donner le pseudo-code de l’algorithme***

Input

Pe : probabilité d’appliquer le cross-over

Pm : probabilité d’appliquer la mutation

Output

Meilleur individu issu du processus génétique

Pop <- generate\_initial\_population() ;

Evaluate(pop) ; //evaluation de la valeur de fitness de chaque individu

While ! StopCondition() do

Parents <- SelectionFrom(pop) ; // typiquement par tournoi binaire

Offspring <- recombine(parents,pe) ; //cross over

Offspring <- mutate(offspring,pm) ;

Evaluate(offspring) ; //remplacement des pires individus de pop par ceux de offspring

insertionInto(pop,offspring) ;

end

return BestIndividualIn(pop) ;

***Question 2 : Expliquer l’utilité des différents paramètres de l’algorithme***

**Individu** : un individu I est un code source composée de modules, correspondant à la définition d’une fonction identifiée par son prototype. Dans la suite, nous faisons l’hypothèse H1 que l’ensemble des modules est défini statiquement pour l’ensemble des individus des populations considérées. Si cela n’empêche pas les modifications internes à chaque module, cette hypothèse permet de garantir la cohérence des opérations de recombinaison d´détaillées par la suite

**Population** : un ensemble de n individus.

**Initialisation de la population** : processus de génération définissant les n premiers individus qui serviront `à alimenter la première génération de l’algorithme. Dans le cadre de cet article, le source `à optimiser est analysé pour en extraire les modules qui le composent. Cette décomposition fournit un individu référence I ref qui est ensuite dérivé en n individus par l’application de mutations aléatoires sur I ref de sorte que l’hypothèse (H1) est respectée.

**Evaluation des individus / ´ fitness** : caractérisation de la qualité d’un individu. Dans le cadre de l’optimisation de code étudiée ici, on s’intéresse au temps d’exécution du code compile sur la machine cible.

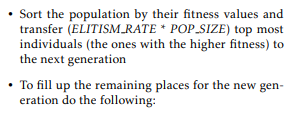
**Sélection** : sélection des individus qui serviront de parents aux opérateurs génétiques (recombinaison et mutation). Plusieurs approches sont possibles (roulette, tournoi etc). Dans cet article, un tournoi binaire est utilisé pour la sélection d’un parent. Il s’agit de tirer au hasard deux individus dans la population et de choisir le gagnant d’un tournoi basé sur la valeur de fitness.

**Recombinaison / Cross-over** : opérateur génétique assurant la création d’un individu I “nouveau-né” `à partir des parents I1 et I2. Bien que l’implémentation de chaque module puisse différer, les types de modules sont communs aux deux parents par l’hypothèse (H1). L’idée consiste alors de composer I de sorte que pour chaque module, un tirage aléatoire et uniforme détermine celui de I1 et I2 qui sera choisi.

**Mutation** : l’opérateur génétique le plus important puisqu’il consiste à générer un nouvel individu I ∗ dérivant de I par l’application des transformations de code mentionnées §2. De fait, ces modifications s’effectuent `a différentes granularités (modules ou sous-modules, instruction). A chaque niveau et sous réserve que ` cela soit possible, une transformation est choisie aléatoirement et est appliquée `à un élément aléatoire de l’individu.

***Question 3 : résolution problème SAT avec l’algorithme générique***

* Population : ensemble de solution généré (random/trivial case …)
* Nous allons effectuer la mutation/crossover sur chaque deux individus de la population et les ajouter à la génération
* Ci-dessus l’algorithme mieux expliqué



O (Algo Gen.) =

NcMax( O (Evaluation) + O (Mutation) + O (Croisement) + O (Selection) )

=NcMax( O(n²) + O(m) + O(2n) + O(n\*m) )

=NcMax(O(n²) )