Algorithm 2 GAFP.

```
1: population = InitializePopulation(populationSize, function);
2: chromosomeLength=GetNumberOfInputs(function);
3: encodedZeroPolarity="";
 4: for (i = 0; i < chromosomeLength; i + +) do
      encodedZeroPolarity+="0";
 6: zeroPolarityExpansion=GenerateFixedPolarity(function.encodedZeroPolarity);
 7: for (i = 0; i < populationSize; i + +) do
      population[i].Expansion=GenerateFixedPolarity(
 8:
           zeroPolarityExpansion,
           population[i].Chromosome);
      population[i].Fitness=Evaluate(population[i].Expansion);
 9:
10: for (i = 0; i < NumberOfIterations; i + +) do
      offsprings=\phi
11:
      parents = RouletteWheelSelection(population);
12:
      for parent_1, parent_2 \in parents do
13:
          (offspring_1, offspring_2) = Crossover(parent_1, parent_2);
14:
          offspring_1 = Mutate(offspring_1);
15:
          offspring_2 = Mutate(offspring_2);
16:
          offspring_1.Expansion = GenerateFixedPolarity(
17:
           zeroPolarityExpansion,
           offspring_1.Chromosome);
18:
          offspring_1.Fitness = Evaluate(offspring_1.Expansion);
19:
          offspring_2.Expansion = GenerateFixedPolarity(
           zeroPolarityExpansion,
           offspring_2.Chromosome);
          offspring_2.Fitness = Evaluate(offspring_2.Expansion);
20:
          offspring = \{offspring\} \cup offspring_1 \cup offspring_2;
21:
22:
      bestSolution = GetBestSolution(offspring);
23:
      for (i = 0; i < ElitismCount; i + +) do
          offspring = \{offspring\} \cup bestSolution;
24:
      population = replace(population, offspring);
25:
26: Return(bestSolution);
```

Algorithm 3 GAMP.

```
1: population = InitializePopulation(populationSize, function);
 2: zeroPolarityExpansion=GenerateMixedPolarity(function,encodedZeroPolarity);
 3: chromosomeLength=
           GetNumberOfInputs(function)*GetNumberOfProductTerms(zeroPolarityExpansion);
 4: encodedZeroPolarity="";
 5: for (i = 0; i < chromosomeLength; i + +) do
       encodedZeroPolarity+="0";
 7: for (i = 0; i < populationSize; i + +) do
       population[i].Expansion=GenerateMixedPolarity(
 8:
           zeroPolarityExpansion,
           population[i].Chromosome);
       population[i].Fitness=Evaluate(population[i].Expansion);
 9:
10: for (i = 0; i < NumberOfIterations; i + +) do
       offsprings=\phi
11:
12:
       parents = RouletteWheelSelection(population);
13:
       for parent_1, parent_2 \in parents do
          (offspring_1, offspring_2) = Crossover(parent_1, parent_2);
14:
          offspring_1 = Mutate(offspring_1);
15:
          offspring_2 = Mutate(offspring_2);
16:
          offspring_1.Expansion = GenerateMixedPolarity(
17:
           zeroPolarityExpansion,
           offspring_1.Chromosome);
          offspring_1.Fitness = Evaluate(offspring_1.Expansion);
18:
          offspring_2.Expansion = GenerateMixedPolarity(
19:
           zeroPolarityExpansion,
           offspring_2.Chromosome);
          offspring_2.Fitness = Evaluate(offspring_2.Expansion);
20:
          offspring = \{offspring\} \cup offspring_1 \cup offspring_2;
21:
       bestSolution = GetBestSolution(offspring);
22:
       for (i = 0; i < ElitismCount; i + +) do
23:
          offspring = \{offspring\} \cup bestSolution;
24:
       population = replace(population, offspring);
25:
26: Return(bestSolution);
```