Remainder stochastic

portion[], Int_portion[], sum_fitness, fitness_value[], average_fitness, POPSIZE, POP[], Temp_pop[], next_pop[], temp_portion[], crossover(), random(Temp_pop[], POPSIZE), string1[], string2[], child[]

```
begin
i = 0, sumfitness = 0;
Repeat
  sumfitness = sumfitness + fitness_value[i];
  i = i + 1;
until( i == POPSIZE );
end;

average_fitness = sum_fitness / POPSIZE;

begin
i = 0;
Repeat
  Portion[i] = fitness_value[i] / average_fitness;
  i = i + 1;
until(i == POPSIZE);
end;

begin
i = 0;
Repeat
  Int_portion[i] = (int)Portion[i];
          {Int_portion[i] = the interger part of portion[i]}
  i = i + 1;
until(i == POPSIZE);
end;

begin
i = 0; N = 0;
repeat
  repeat
```

```
    Temp_pop[N] = POP[i]
    N = N + 1;
  until(N == Int_portion[i]);
  i = i + 1;
until(i == POPSIZE);
select  **N**;
end;

begin
i = 0;
repeat
temp_portion[i] = portion[i] – Int_portion[i];
until(i == POPSIZE);
end;

sort(temp_portion[i]); {sort index (i) from the largest to the smallest}

begin
i = 0; (sorted index i)
repeat
Temp_pop[**N**] = POP[i];
n = n + 1, i = i + 1;
until(N == POPSIZE);
end;

begin
i = 0;
repeat
  string1[] = random(Temp_pop[], POPSIZE);
  string2[] = random(Temp_pop[], POPSIZE);
  child[i] = crossover(string1[], string2[]);
  next_pop[i] = child[i];
  i = i + 1;
until(i == POPSIZE);
end;
```