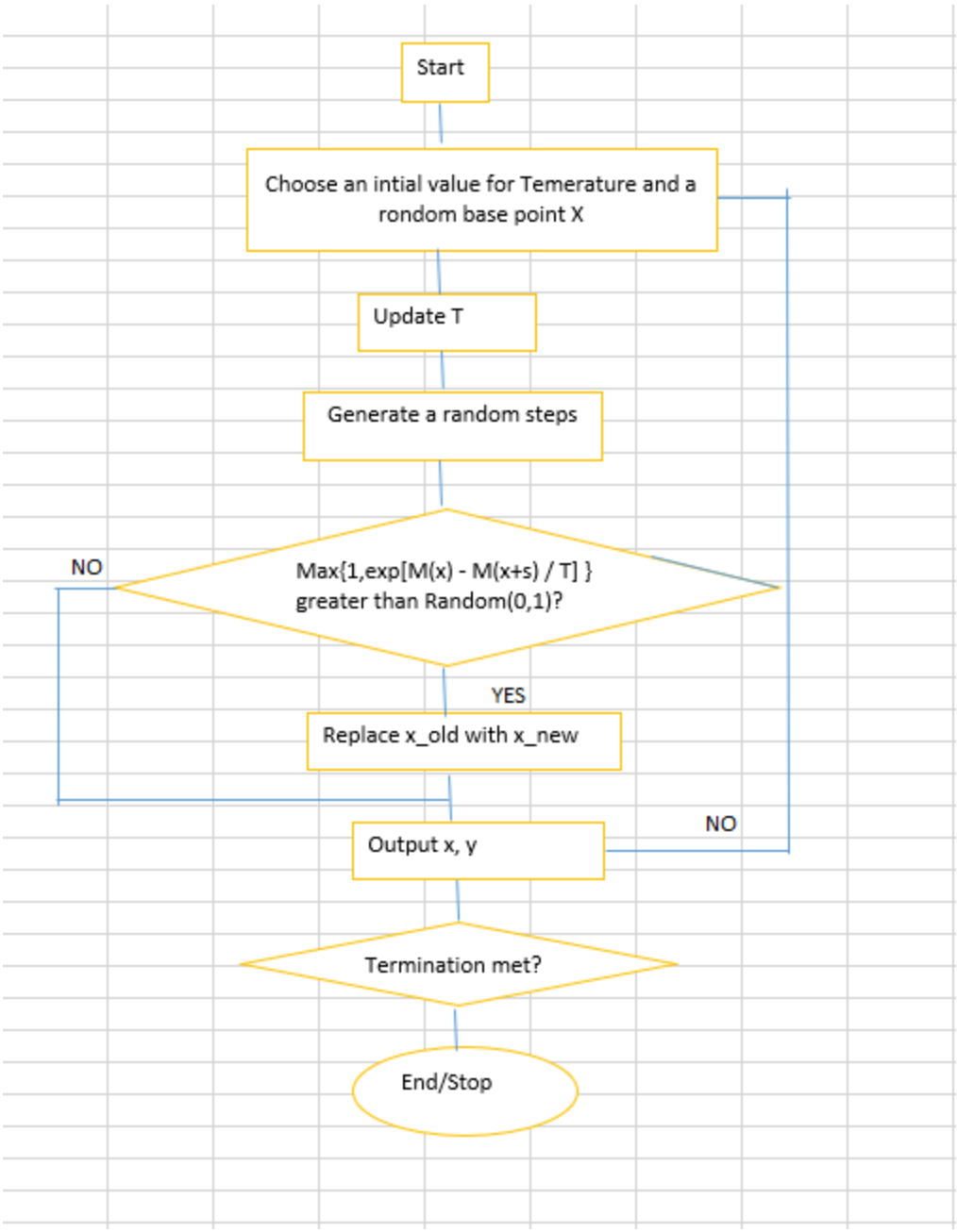
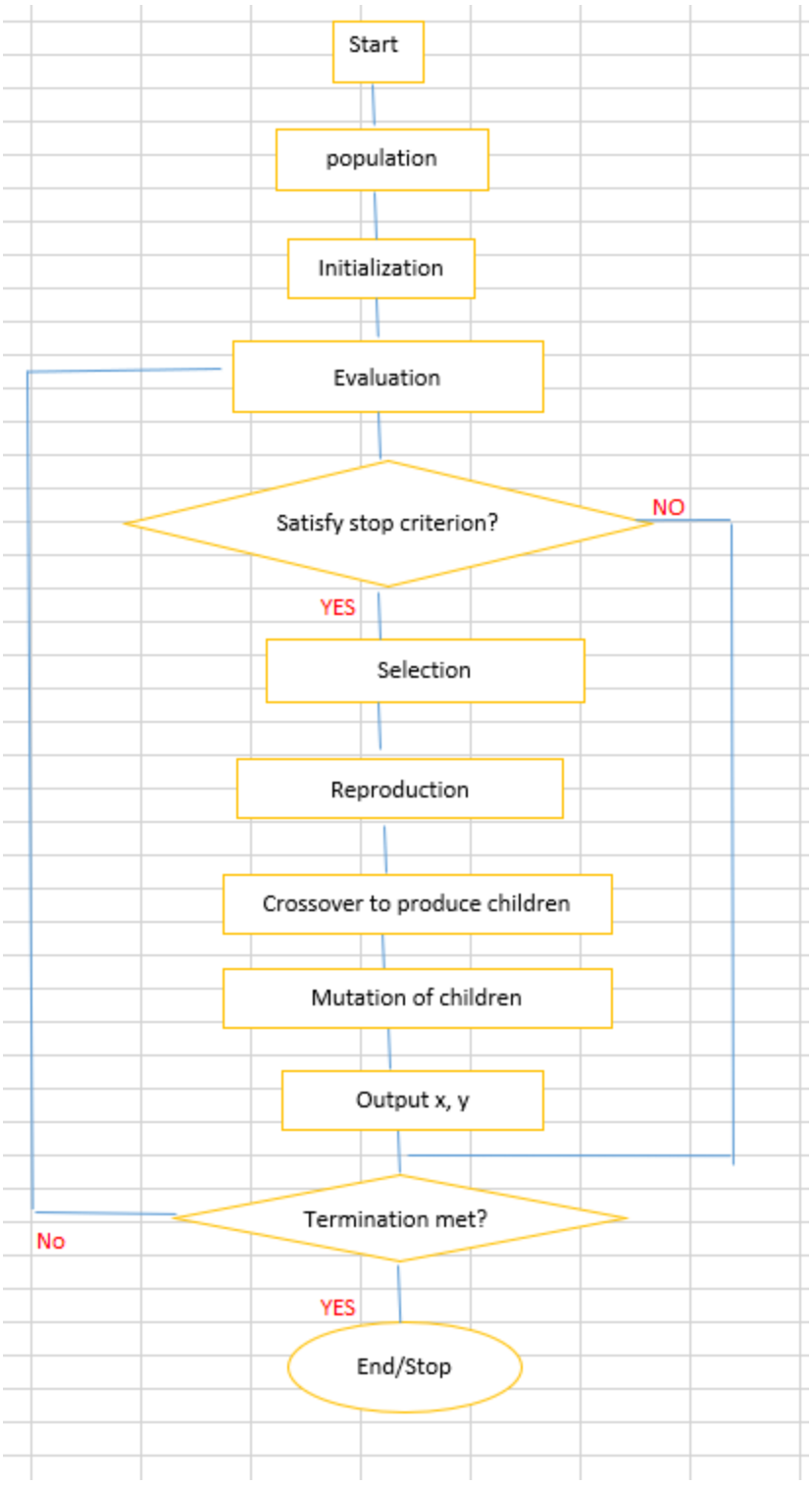
Simulated Annealing - Pseudo-Code

```
#Finding the maximum point of y = 10 * math.sin(5 * x) + 7 * math.cos(4 * x)

#Parameter Set
low = 0
high = 9
tmp = 1e5
tmp_min = 1e-3
alpha = 0.95

#Initilization
x_old = 0
x_new = x_old
value_old = ObjFun(x_old)
value_new = value_old

counter = 0
record_x = []
record_y = []


#Define Fucntion
def ObjFun(x):
    y = 10 * math.sin(5 * x) + 7 * math.cos(4 * x)

#Change_temperature
while(tmp > tmp_min and counter <= 100000):
    deltaE = value_new - value_old
    value_new = ObjFun(x_new)
    if deltaE >=0:
        return value_best = value_new
    if exp(deltaE/tmp)>random(0,1):
        return value_best = value_new
    else deltaE < 0:
        tmp=tmp*alpha
    endif
        counter+=1

```


Start
Choose an intial value for Temerature and a rondom base point X
Update T
Generate a random steps
Max{1,exp[M(x) - M(x+s) / T] } greater than Random(0,1)?
NO
YES
Replace x_old with x_new
Output x, y
NO
Termination met?
End/Stop

Genetic Algorithms - Pseudo-Code

```
#Finding the maximum point of y = 10 * math.sin(5 * x) + 7 * math.cos(4 * x)

#Parameter Set
cross_rate = 0.75
mute_rate = 0.15

#Initilization
x = an individual number within P
P = generate a population of individuals randomly within [-10,10]
y1 = ObjFun(child1)
y2 = ObjFun(child2)
counter = 10000

#Define Fucntion
def ObjFun(x):
    y = 10 * math.sin(5 * x) + 7 * math.cos(4 * x)

#Modification
while (x is within [-10,10]):
    while size(P') < size(P)
        parent1 <- number_selection(P)
        parent2 <- number_selection(P)
        child1, child2 <- with probability cross_rate 0.75 crossover parent1, parent2
        child1 <- mutate child1 with mutate rate 0.15
        child2 <- mutate child2 with mutate rate 0.15
        add child1 and child 2 to population P'

    elitism: if the best fitness individual is not in P'',
    replace the worst individual in P'' with this best one
    P <- P'
    P' <- ()

end while
    return child1,child2,y1,y2
```


Start
population
Initialization
Evaluation
Satisfy stop criterion?
NO
YES
Selection
Reproduction
Crossover to produce children
Mutation of children
Output x, y
Termination met?
No
YES
End/Stop

Simulated Annealing - Pseudo-Code （WORD Version）

```
#Parameter Set
low = 0
high = 9
tmp = 1e5
tmp_min = 1e-3
alpha = 0.95

#Initilization
x_old = 0
x_new = x_old
value_old = ObjFun(x_old)
value_new = value_old

counter = 0
record_x = []
record_y = []



#Define Fucntion
def ObjFun(x):
   y = 10 * math.sin(5 * x) + 7 * math.cos(4 * x)

#Change_temperature
while(tmp > tmp_min and counter <= 100000):
   deltaE = value_new - value_old
   value_new = ObjFun(x_new)
   if deltaE >=0:
      return value_best = value_new
```

```
   if exp(deltaE/tmp)>random(0,1):
      return value_best = value_new
   else deltaE < 0:
      tmp=tmp*alpha
   endif
      counter+=1
```

Genetic Algorism - Pseudo-Code （WORD Version）

```
#Finding the maximum point of y = 10 * math.sin(5 * x) + 7 * math.cos(4 * x)

#Parameter Set
cross_rate = 0.75
mute_rate = 0.15

#Initilization
x = an individual number within P
P = generate a population of individuals randomly within [-10,10]
y1 = ObjFun(child1)
y2 = ObjFun(child2)
counter = 10000

#Define Fucntion
def ObjFun(x):
   y = 10 * math.sin(5 * x) + 7 * math.cos(4 * x)

#Modification
while (x is within [-10,10]):
          while size(P') < size(P)
```

parent1 <- number_selection(P)

parent2 <- number_selection(P)

child1, child2 <- with probability cross_rate 0.75 crossover parent1, parent2

child1 <- mutate child1 with mutate rate 0.15

child2 <- mutate child2 with mutate rate 0.15

add child1 and child 2 to population P'

elitism: if the best fitness individual is not in P'', replace the worst individual in P'' with this best one

P <- P'

P' <- ()

end while

return child1,child2,y1,y2