```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#define maxpop 200
#define F  0.9
#define CR 0.3
#define genmax 1000
typedef struct tind
{
  float x[11];
  float fitness;
} ind;
ind newpopulation[maxpop],oldpopulation[maxpop];
float p[8],init[11] ={ 0,0,0,0,0,0,85,90,3,1.2,145},end[11]={0,2000,16000,120,5000,2000,93,95,12,
4,162},fval;
float mod    ( float f)
{
    if (f<0) return -1*f ;else return f;
}
int notdifferent ( int index1,int index2,int index3,int i)
{
    if ( index1 == index2 || index1== index3 || index1==i)
    return 1;
    else if ( index2 == index3 || index2 == i)
    return 1;
    else if ( index3 == i) return 1;
    else return 0;
}
float isbounded( ind i)
{
 float diff=0;
 int j;
 for (j=1;j<11;++j)
 {
    if (i.x[j]> end[j])
        diff+=i.x[j]-end[j];
    if (i.x[j]<init[j])
        diff+=init[j]-i.x[j];
 }
 return 100*diff;
}



float constraints ( ind i)
{
    float penalty=0,boundpen,d4l=0.99,d4u=1.01,d7l=0.99,d7u=1.01,d9l=0.9,d9u=1.111,d10l=0.99,d10u
=1.01;
    p[0]=i.x[1]*(1.12+.13167*i.x[8]-0.00667*i.x[8]*i.x[8])-d4l*i.x[4];
    if (p[0]<0)  penalty+=10*mod(p[0]);
    p[1]=i.x[1]*(1.12+.13167*i.x[8]-0.00667*i.x[8]*i.x[8])-d4u*i.x[4] ;
    if (p[1] > 0) penalty+=10*p[1];
    p[2]=86.35+1.098*i.x[8]-0.038*i.x[8]*i.x[8]+0.325*(i.x[6]-89)-d7l*i.x[7];
    if (p[2]<0) penalty+=10*mod(p[2]);
    p[3]=86.35+1.098*i.x[8]-0.038*i.x[8]*i.x[8]+0.325*(i.x[6]-89)-d7u*i.x[7] ;
    if (p[3]>0) penalty+=10*p[3];

    p[4]=35.82-0.222*i.x[10]-d9l*i.x[9];
    if (p[4]<0) penalty+=10*mod(p[4]);
    p[5]=35.82-0.222*i.x[10]-d9u*i.x[9] ;
    if (p[5]>0) penalty+=10*p[5];
    p[6]=(-133+3*i.x[7])-d10l*i.x[10];
     if (p[6]<0) penalty +=10*mod(p[6]);
    p[7]=(-133+3*i.x[7])-d10u*i.x[10];
    if (p[7]>0) penalty +=10*p[7];
    boundpen=10*isbounded(i);
    return (12*penalty+boundpen);
}
float fitness ( ind i)
{
    float penalty,c1=0.063,c2=5.04,c3=0.035,c4=10,c5=3.36;
    penalty=constraints(i);
    fval = c1*i.x[4]*i.x[7]-c2*i.x[1]-c3*i.x[2]-c4*i.x[3]-c5*i.x[5];
    return (c1*i.x[4]*i.x[7]-c2*i.x[1]-c3*i.x[2]-c4*i.x[3]      -c5*i.x[5]-penalty);
}
ind mutate ( int index1, int index2,int index3)
```

```
{
    ind new;
        int i;
    for ( i=1;i<=10;++i)
        new.x[i]=oldpopulation[index3].x[i] + F*mod((oldpopulation                          [index2
].x[i]-oldpopulation[index1].x[i]));
    new.fitness=fitness(new);
    return new;
}
 int isindependent( int i)
 {
    if ( i==5 || i==6 || i==2 ) return 0; else return 1;
 }

 void  initialise (int popsize)
 {
  int i,j,temp;
  for ( i=0;i<popsize;++i)
  {
    for (j=1;j<11;++j)
        if (isindependent(j)) oldpopulation[i].x[j]=init[j]+(end[j]-init[j])*drand48();
        oldpopulation[i].x[5]=1.22*oldpopulation[i].x[4]-oldpopulation[i].x[1];
        oldpopulation[i].x[6]=98000*oldpopulation[i].x[3]/(oldpopulation[i].x[4]*oldpopulation[i]
.x[9]+1000*oldpopulation[i].x[3]);
        oldpopulation[i].x[2]=oldpopulation[i].x[8] * oldpopulation[i].x[1]- oldpopulation[i].x[5
];


    printf ( "here");
    oldpopulation[i].fitness= fitness (oldpopulation[i]);
  }
}
ind cross (ind oldpop,ind newpop)
{
    ind new;
    int i;
    float r;
    for (i=1;i<11;++i)
    {
        if (!isindependent(i)) continue;
        r=drand48();
        if (r>CR) new.x[i]=oldpop.x[i] ;
        else new.x[i]=newpop.x[i];
        if ( CR==0 && i==10) new.x[i]=newpop.x[i];
    }
        new.x[5]=1.22*new.x[4]-new.x[1];
        new.x[6]=98000*new.x[3]/(new.x[4]*new.x[9]+1000*new.x[3]);
        new.x[2]=new.x[8]*new.x[1]-new.x[5];
    new.fitness=fitness(new);
    return new;
}

void swap(int popsize)
{
    int i,j;
    for (i=0;i<popsize;++i)
    {
        for ( j=1;j<11;++j)
        oldpopulation[i].x[j]=newpopulation[i].x[j];
        oldpopulation[i].fitness=newpopulation[i].fitness;
    }
}
float actval ( int i)
{
    switch(i)
    {
        case 1:
        return 1698.1;
        case 2:
        return 15819;
        case 3:
        return 54.107;
        case 4:
        return 3031.2;
        case 5:
        return 2000;
```

```
        case 6:
        return 90.115;
        case 7:
        return 95;
        case 8:
        return 10.493;
        case 9:
        return 1.5618;
        case 10:
        return 153.54;
    }
}



int printsol(int popsize)
{
 int i,j;
 float xval[20],sds=0,sd,temp,meanfit=0,fit[20],sumx[11]={0,0,0,0,0,0,0,0,0,0,0},fval1=0,meanpen=
0,pen[20];
 static float sfit=0;
 for (i=0;i<popsize;++i)
 {
    fit[i] =fitness(oldpopulation[i]);
    pen[i] = constraints(oldpopulation[i]);
    meanfit+=fit[i];
    meanpen+=pen[i];
    fval1+=fval;
        for (j=1;j<=10;++j)
         sumx[j]+=oldpopulation[i].x[j];
 }
    meanfit/=popsize;
    meanpen/=popsize;
    fval1/=popsize;

    if ( sfit )
    if (mod(meanfit-sfit) <= .001*sfit)
    {


            printf("\nOBJECTIVE FUNCTION =  %f MEAN PENALTY = %f\n",fval1,meanpen);
            for (j=1;j< 11;++j)
            {
            xval[j]=sumx[j]/popsize;
            printf(" x[%i] = %f (%f)\t",j,xval[j],actval(j));
            temp=(xval[j]-actval(j))/actval(j);
            temp*=temp;
            sds+=temp;
            }
            sds/=9;
            sd=sqrt(sds);
            printf("\n\nTHE SD IS  %f\n\n",sd);
            for ( j=0;j<8;++j)
            printf(" %i constraint is  %f\t",j+1,p[j]);

        return 1 ;
    }
        sfit=meanfit;
        return 0;
}

main()
{
 ind trialind,newind;
 int c,gen,i=0,popsize=4,index1,index2,index3;
 float r1,r2,r3;
 while ( ++popsize <= 100)
 {
 printf("***********************************************\n");
 printf("\nSTARTING WITH POPSIZE = %i\n",popsize);
 initialise ( popsize);
 for ( gen=0;gen<genmax;++gen)
 {

 if ( gen!=0)
  swap(popsize);
```

```
 if ( (gen+1) % 100==0)
 {
 c=printsol(popsize);
 if ( c==1)
 {
  printf("\n\nCONVERGED AFTER %i GENERATIONS\n",gen+1);
  gen=genmax;
 }

 }
 for (i=0;i<popsize;++i)
 {
 do
 {
 r1=drand48();
 index1=r1*(popsize-1);
 r2=drand48();
 index2=r2*(popsize-1);
 r3=drand48();
 index3=r3*(popsize-1);
 }while (notdifferent ( index1,index2,index3,i));
 newind= mutate(index1,index2,index3);
 trialind= cross(oldpopulation[i],newind);
 if ( trialind.fitness > oldpopulation[i].fitness )
 newpopulation[i]=trialind;
 else
 newpopulation[i]=oldpopulation[i];
 }
 }
 }
}
```