Enoncé 1 : Donner l'affichage lors de l'exécution du programme suivant ainsi qu'un schéma décrivant le stockage dans la mémoire des objets et attributs correspondants.

```
class vect;
class matrice {
    double mat[3][3];
    public:
        matrice (double t[3][3])   {      int i, j;
                for(i=0;i<3;i++)
                        for(j=0;j<3;j++)
                                mat[i][j] = t[i][j];
                                }
        vect prod(vect); // fonction membre
};
class vect {
      double v[3]; static int nb;
      public :
        vect (double v1=0,double v2=0,double v3=0){
                v[0]=v1; v[1]=v2; v[2]=v3;                          nb++;
                cout <<" ----VEC CONSTR -----"<< nb << "\n";    }
        vect (vect const &vs) : vect(vs.v[0], vs.v[1], vs.v[2]) {
                cout << "***** Copy is done : " << nb << " ***** \n";  }
        friend vect matrice::prod(vect);
        void affiche(){
                for(int i=0;i<3;i++)       cout<< "v: " << v[i]<<"\t" ;
                cout << endl;    }
        ~vect()  {  --nb; cout << " __ deleted object "<< nb << " ___\n";                          }
        };
int vect::nb=0;
vect matrice::prod(vect x){
        int i,j; double som;
                vect vr; cout << " \t OK \n"; //pour le résultat
                        for (i=0;i<3;i++) {
                                som = 0;
                                for (j=0;j<3;j++)
                                        som += mat[i][j]*x.v[j];
                                vr.v[i] = som; }
        return vr;        }

int main() {
        vect w(1,2,3);
        vect* res = new vect();
        double tb[3][3]={1,2,3,4,5,6,7,8,9};
        matrice a = tb;                                 *res= a.prod(w);
        cout << " \t NOT YET \n";                        res->affiche();
        vect r(*res);     delete res;       r.affiche();
        return 0; }
```