SOLUCIÓN PROBLEMA 2

```
#include <stdio.h>
#include <stdlib.h>

#define N 6
#define M 6

void esconder_tesoro(float mapa[N][M]);
void buscar_tesoro(float mapa[N][M], int *fila, int *columna);
void distancia_tesoro(float mapa[N][M], int fila, int columna);
void imprimir_mapa(float mapa[N][M]);
char usar_radar(int tipo, float mapa[N][M], int fila, int columna);

int main(int argc, char *argv[])
{

  float mapa[N][M];
  int n_intentos = 0; // control del número de intentos
  int f, c; // coordenadas de exploración
  int radar = 0; // control de utilización del radar
  char orientacion; // almacenamiento de la orientación del tesoro respecto de las coordenadas de exploración

  esconder_tesoro(mapa);

  do{
       // lectura de coordenadas de exploración
       printf("\nIntento %d de %d",n_intentos + 1,(N + M) / 3);
       printf("\nCoordenadas de busqueda (mapa de %dx%d):\n",N, M);
       scanf("%d%d",&f,&c);

       // cálculo de la distancia euclídea al tesoro desde las coordenadas de exploración
       distancia_tesoro(mapa, f, c);

       if(mapa[f][c] != 0){ // si no se ha encontrado el tesoro

                    imprimir_mapa(mapa);

                    if(radar == 0){ // si aún no se ha utilziado el radar, le damos la opción al usuario
                        printf("\nQuieres usar el radar? ");
                        scanf("%d", &radar);
                        if(radar == 1){
                                 printf("\nElige un radar, horizontal (1) o vertical (0): ");
                                 scanf("%d",&radar); // reutilizamos 'radar'para el tipo
                                 orientacion = usar_radar(radar, mapa, f, c);

                                 switch(orientacion){
                                     case 'N': printf("\nEl tesoro se encuentra hacia el NORTE."); break;
                                     case 'S': printf("\nEl tesoro se encuentra hacia el SUR."); break;
                                     case 'E': printf("\nEl tesoro se encuentra hacia el ESTE."); break;
                                     case 'O': printf("\nEl tesoro se encuentra hacia el OESTE."); break;
                                     default:
                                              if(radar == 1)
                                                       printf("\nEl  tesoro  se  encuentra  en  tu  misma
longitud.");
```

```
                                        else
                                                printf("\nEl tesoro se encuentra en tu misma latitud.");
                                }
                                 radar = -1; // ya no se puede usar el radar otra vez
                            }
                        }
        }

        n_intentos++;

   }while((mapa[f][c] != 0) && (n_intentos < (N + M) / 3)); // leemos coordenadas de exploración hasta que el usuario encuentre el tesoro o se supere el número máximo de intentos

   if(mapa[f][c] == 0){
           printf("\nHAS GANADO!!\nLo has conseguido en %d intentos de %d posibles\n", n_intentos, (N + M) / 3);
   }else{
           printf("\nHas superado el numero de intentos permitidos (%d).", (N + M) / 3);
   }

   system("PAUSE");
   return 0;
}

// FUNCIÓN PROPORCIONADA
void esconder_tesoro(float mapa[N][M]){

     int i, j;
     int f = 0, c = 0;

     srand(time(NULL));
     f = rand() % (N);
     c = rand() % (M);

     for(i = 0; i < N; i++){
           for(j = 0; j < M; j++){
                 mapa[i][j] = 0;
           }

     }
     mapa[f][c] = -1;
}

void buscar_tesoro(float mapa[N][M], int *fila, int *columna){

     int encontrado = 0; // control de búsqueda, para no hacer iteraciones innecesarias

     *fila = 0;
     while((encontrado == 0) && (*fila < N)){
             *columna = 0;
             while((encontrado == 0) && (*columna < M)){
                  if(mapa[*fila][*columna] == -1) // el tesoro está representado por un '-1' en la matriz mapa
                          encontrado = 1;
                  else // sólo incrementamos la columna si el tesoro no ha sido encontrado
                          *columna = *columna + 1;
```

```
            }
            if(encontrado == 0) // sólo incrementamos la fila si el tesoro no ha sido encontrado
                 *fila = *fila + 1;
     }
}

void distancia_tesoro(float mapa[N][M], int fila, int columna){

      int f_tesoro, c_tesoro;

      buscar_tesoro(mapa, &f_tesoro, &c_tesoro);

      mapa[fila][columna] = sqrt(pow(f_tesoro - fila,2) + pow(c_tesoro - columna,2)); // cálculo de la distancia
euclídea al tesoro



}

void imprimir_mapa(float mapa[N][M]){
     int i, j, f_tesoro, c_tesoro;

     buscar_tesoro(mapa, &f_tesoro, &c_tesoro);

     printf("\n");
     for(i = 0; i < N; i++){
           printf("%d |   ",i);
           for(j = 0; j < M; j++){
                 if((i == f_tesoro) && (j == c_tesoro)) // mostramos las coordenadas del tesoro como terreno
sin explorar
                       printf("%.2f\t",0);
                 else
                       printf("%.2f\t",mapa[i][j]);
           }
           printf("\n\n");
     }

     for(i = 0;i < M; i++){
           printf("\t----");
     }
     printf("\n");
     for(i = 0;i < M; i++){
           printf("\t  %d",i);
     }

}

char usar_radar(int tipo, float mapa[N][M], int fila, int columna){

    int f_tesoro, c_tesoro;

    buscar_tesoro(mapa, &f_tesoro, &c_tesoro);

    switch(tipo){
                 case 1: // radar horizontal
                      if(c_tesoro == columna) // igual longitud
```

```
                    return 'L';
               else if(c_tesoro > columna)
                    return 'E'; // Este
               else
                    return 'O'; // Oeste
          break;
          case 2: // radar vertical
               if(f_tesoro == fila) // igual latitud
                    return 'L';
               else if(f_tesoro > fila)
                    return 'S'; // Sur
               else
                    return 'N'; // Norte
          break;

     }
}
```