```c


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

enum {FALSE = 0, TRUE = !0};
typedef unsigned int Bool;

Bool EsFactible (int c, int p, int acum)
{
    if ((c+acum) <= p) {
        return TRUE;
    }
    else {
        return FALSE;
    }
}

Bool EsSolucion (int p, int acum)
{
    if (acum == p) {
        return TRUE;
    }
    else {
        return FALSE;
    }
}



Bool bt (int c[], int i, int j, int s[], int* s_idx, int p, int* acum)
{
    Bool success = FALSE;
    // termina cuando encuentre una solución

    for (int x = i; x < j; ++x) {

        if (success == TRUE) { return TRUE; }
        // ya no revisa los candidatos restantes una vez que se obtuvo una
        // solución

        printf ("\nProbando al candidato <%d>\n", c[x]);

        if (EsFactible (c[x], p, *acum)) {

            // registra al candidato:
            int cand = c[x];
            s[*s_idx] = cand;
            ++*s_idx;
            *acum += cand;

            printf ("Registrando al candidato <%d>\n", cand);

            if (!EsSolucion (p, *acum)) {
                success = bt (  c, x+1, j,
                                s, s_idx,
                                p, acum);

                if (success == FALSE) {

                    // cancela el registro:
```

```
62                      --*s_idx;
63                      *acum -= cand;
64
65                      printf ("Borrando al candidato <%d>\n", cand);
66                  }
67              }
68              else {
69                  printf ("Se encontró una solución!!\n");
70                  return TRUE;
71              }
72          }
73          else {
74              printf ("El candidato <%d> no es factible\n", c[x]);
75          }
76      }
77
78      printf ("Se terminaron los candidatos\n");
79      return FALSE;
80  }
81
82
83
84  /*--------------------------------------------------------------------------*/
85  /**
86   * @brief Función de activación
87   *
88   * @param c[] Conjunto de candidatos
89   * @param tam_c Número de candidatos
90   * @param s[] Conjunto solución
91   * @param tam_s Número máximo de elementos en el conjunto solución
92   * @param p Instancia particular del problema
93   *
94   * @return El tamaño del conjunto solución; si es cero, entonces no se encontró
95   * ninguna solución
96   */
97  /*--------------------------------------------------------------------------*/
98  int Cambio (int c[], int tam_c, int s[], int tam_s, int p)
99  {
100     int acum = 0;
101     int s_idx = 0;
102
103     Bool res = bt ( c, 0, tam_c, // envía al conjunto de candidatos completo
104                     s, &s_idx,  // conjunto solución
105                     p,          // instancia particular del problema
106                     &acum);     // acumulador
107
108     return s_idx;
109 }
110
111
112 int main(void)
113 {
114     int candidatos[5] = {1, 2, 5, 5, 3};
115
116     int solucion[5] = {};
117     // a lo más, habrá 5 monedas
118
119     int p = 10;
120
121     int res = Cambio (candidatos, 5, solucion, 5, p);
122
```

```
123         if (res > 0) {
124             printf ("La solución es: \n");
125             for (int i = 0; i < res; ++i) {
126                 printf ("%d\n", solucion[i]);
127             }
128         }
129         else {
130             printf ("No se encontró una solución\n");
131         }
132 
133         return 0;
134     }
135 
136 
```