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

#define true 1
#define false 0

typedef int bool;

typedef struct {
  double peso;
  double valor;
} ITEM;

ITEM criarItem(double p, double v){
  ITEM novo;
  novo.peso = p;
  novo.valor = v;
  return novo;
}

/* Implementacao gulosa para a solucao do problema da mochila fracionada.
   Esta implementacao assume que os itens estao ordenados pelo melhor relacao
   entre valor e peso (de forma decrescente) para produzir a solucao otima. */
double mochilaFracionadaG(ITEM itens[], int n, double capacidade) {
  int i;
  double carga = 0;
  double valor = 0;
  double fracao;
  for (i=0; i<n; i++)
    if (carga+itens[i].peso <= capacidade) {
      valor += itens[i].valor;
      carga += itens[i].peso;
    }else{
      fracao = (capacidade - carga)/itens[i].peso;
      valor += itens[i].valor*fracao;
      carga = capacidade;
      break;
    }
  return valor;
}
```

```
/* Implementacao gulosa para a solucao do problema da mochila
binaria.
   Nao ha garantia de solucao otima, independente da ordenacao dos
itens. */
double mochilaBinariaG(ITEM itens[], int n, double capacidade) {
  int i;
  double carga = 0;
  double valor = 0;
  for (i=0; i<n; i++)
    if (carga+itens[i].peso <= capacidade) {
      valor += itens[i].valor;
      carga += itens[i].peso;
    }
  return valor;
}


double maiorValor = 0;
double valorAtual;

/* Implementacao usando Tentativa e Erro para a solucao do problema
   da mochila binaria.
   Encontrara a melhor solucao, independentemente da ordem dos
itens. */
void mochilaBinariaTEAux(ITEM itens[], int n, double capacidade, int
atual) {
  if (atual == n) {
    if (valorAtual > maiorValor) maiorValor = valorAtual;
    return;
  }
  /* tenta: vai para o proximo item sem colocar o atual na mochila
*/
  mochilaBinariaTEAux(itens, n, capacidade, atual+1);
  /* na volta tenta, se viavel, colocar o item atual na mochila */
  if (capacidade>=itens[atual].peso){
    valorAtual+=itens[atual].valor;
    /* parte da mudanca de estado eh passada como parametro,
diminuindo a
       capacidade (disponivel) da mochila */
    mochilaBinariaTEAux(itens, n, capacidade-itens[atual].peso,
atual+1);
    valorAtual-=itens[atual].valor;
  }
}

double mochilaBinariaTentativaEErro(ITEM itens[], int n, double
capacidade) {
  maiorValor = 0;
  valorAtual = 0;
  mochilaBinariaTEAux(itens, n, capacidade, 0);
  return maiorValor;
}
```

```c
/* Funcao main que cria uma instancia do problema da mochila e
   chama a funcao gulosa e usando tentativa e erro para resolver.
   Os itens estao ordenados de forma descrescente de acordo com a
relacao
   peso/valor.
*/
int main() {

  /* Os itens estao ordenados pela relacao valor/peso de forma
decrescente. */
  ITEM itens[3];
  itens[0] = criarItem(10,  60);
  itens[1] = criarItem(20, 100);
  itens[2] = criarItem(30, 120);

  printf("Executando o algoritmo guloso para solucao do problema da
mochila fracionada.\n");
  printf("Valor colocado dentro da mochila (20kg): %.2f\
n",mochilaFracionadaG(itens,3,20));

  printf("\nExecutando o algoritmo guloso para solucao do problema da
mochila fracionada.\n");
  printf("Valor colocado dentro da mochila (50kg): %.2f\
n",mochilaFracionadaG(itens,3,50));

  printf("\nExecutando o algoritmo guloso para solucao do problema da
mochila fracionada.\n");
  printf("Valor colocado dentro da mochila (60kg): %.2f\
n",mochilaFracionadaG(itens,3,60));


  printf("\n\nExecutando o algoritmo guloso para solucao do problema
da mochila binaria.\n");
  printf("Valor colocado dentro da mochila (20kg): %.2f\
n",mochilaBinariaG(itens,3,20));

  printf("\nExecutando o algoritmo guloso para solucao do problema da
mochila binaria.\n");
  printf("Valor colocado dentro da mochila (50kg): %.2f\
n",mochilaBinariaG(itens,3,50));

  printf("\nExecutando o algoritmo guloso para solucao do problema da
mochila binaria.\n");
  printf("Valor colocado dentro da mochila (60kg): %.2f\
n",mochilaBinariaG(itens,3,60));


  printf("\n\nExecutando o algoritmo de Tentativa e Erro para solucao
do problema da mochila binaria.\n");
  printf("Valor colocado dentro da mochila (20kg): %.2f\
n",mochilaBinariaTentativaEErro(itens,3,20));
```

```
  printf("\nExecutando o algoritmo de Tentativa e Erro para solucao
do problema da mochila binaria.\n");
  printf("Valor colocado dentro da mochila (50kg): %.2f\
n",mochilaBinariaTentativaEErro(itens,3,50));

  printf("\nExecutando o algoritmo de Tentativa e Erro para solucao
do problema da mochila binaria.\n");
  printf("Valor colocado dentro da mochila (60kg): %.2f\
n",mochilaBinariaTentativaEErro(itens,3,60));
  return 0;
}

/* SAIDA

Executando o algoritmo guloso para solucao do problema da mochila
fracionada.
Valor colocado dentro da mochila (20kg): 110.00

Executando o algoritmo guloso para solucao do problema da mochila
fracionada.
Valor colocado dentro da mochila (50kg): 240.00

Executando o algoritmo guloso para solucao do problema da mochila
fracionada.
Valor colocado dentro da mochila (60kg): 280.00


Executando o algoritmo guloso para solucao do problema da mochila
binaria.
Valor colocado dentro da mochila (20kg): 60.00

Executando o algoritmo guloso para solucao do problema da mochila
binaria.
Valor colocado dentro da mochila (50kg): 160.00

Executando o algoritmo guloso para solucao do problema da mochila
binaria.
Valor colocado dentro da mochila (60kg): 280.00


Executando o algoritmo de Tentativa e Erro para solucao do problema
da mochila binaria.
Valor colocado dentro da mochila (20kg): 100.00

Executando o algoritmo de Tentativa e Erro para solucao do problema
da mochila binaria.
Valor colocado dentro da mochila (50kg): 220.00

Executando o algoritmo de Tentativa e Erro para solucao do problema
da mochila binaria.
Valor colocado dentro da mochila (60kg): 280.00
*/
```