問題4

ナップサック問題を解く以下の C プログラムについて問に答えよ．ナップサック問題とは，自然数 N と実数 W，$(c_i)_{i=0..N-1}$，$(w_i)_{i=0..N-1}$が与えられたときに，$\sum_{i=0}^{N-1} p_i \cdot w_i \leq W$の制約下，$\sum_{i=0}^{N-1} p_i \cdot c_i$を最大化する$(p_i)_{i=0..N-1}$ $(p_i = 0$ 又は $1)$を求める問題である．直感的にはWはナップサックに詰められる重さの限界，$(c_i)_{i=0..N-1}$は各品物の価値，$(w_i)_{i=0..N-1}$は各品物の重さを表し，問題はナップサックに詰める品物の価値の総和を最大化することである．以下，簡単のため $c_i > 0\,(0 \leq i < N)$，$w_i > 0\,(0 \leq i < N)$，$\frac{c_i}{w_i} \geq \frac{c_j}{w_j}\,(0 \leq i < j < N)$ を仮定する．

```
1: double max = -1.0, c[N], w[N];
2: int tmp[N], p[N];
3: void knapsack(int i, double sum, double weight)
4: {
5:   double z, relaxedks();
6:   if (i == N) {
7:     if (sum > max) {
8:       max = sum;
9:       for (int j = 0; j < N; j++) p[j] = tmp[j];
10:    }
11:  } else {
12:    if (w[i] <= weight) {
13:      tmp[i] = 1;
14:      knapsack(i+1, sum+c[i], weight-w[i]);
15:    }
16:    z = relaxedks(i+1, weight);
17:    if (z+sum > max) {
18:      tmp[i] = 0;
19:      knapsack(i+1, sum, weight);
20:    }
21:  }
22:}
23:double relaxedks(int i, double weight)
24:{
25:  int j;
26:  double r = 0.0;
27:  for (j = i; j < N; j++) {
28:    if (w[j] < weight) {
29:      r += c[j]; weight -= w[j];
30:    } else return r + weight * c[j]/w[j];
31:  }
32:  return r;
33:}
34:void main()
35:{
36:  knapsack(0,0.0, W);
37:}
```

(1) N=5, W=8.0, $(c_i)_{i=0..4}$ = (3.0, 4.0, 7.0, 5.0, 7.0), $(w_i)_{i=0..4}$ = (1.0, 2.0, 4.0, 3.0, 5.0) であるとき，解となる$(p_i)_{i=0..4}$をひとつ与えよ．

(2) プログラムの N, W, c[], w[]を問(1) のように設定して関数 main を実行したとき，終了するまでの関数 knapsack の呼び出しを，順番に，引数とともに列挙せよ．

(3) $(p_i)_{i=0..N-1}$として実数　$0 \le p_i \le 1$ を取れるように条件を緩和する．このときに，$\sum_{i=0}^{N-1} p_i \cdot w_i \le W$ の制約下，$\sum_{i=0}^{N-1} p_i \cdot c_i$ を最大化する $(p_i)_{i=0..N-1}$ を求めよ．関数 relaxedks を参照せよ．

(4) プログラム 17−20 行目の条件文は，条件（z+sum > max）が真であるときに限り，関数 knapsack がさらに呼び出される可能性のあることを示している．これで解が求められる理由を説明せよ．

(5) プログラムが，解のうち少なくとも 1 つを実際に検査し，結果としてナップサック問題の解の 1 つを与えることを説明せよ．

(6) max の初期値を 0.0 とすると，プログラム中，解を p[]に代入する部分にあたる 9 行目が 1 度も実行されない場合がある．そのような場合を特定し，理由とともに述べよ．

Problem 4

Answer the questions on the following C program that solves the knapsack problem. The knapsack problem is specified as: given a natural number N and real numbers W, $(c_i)_{i=0..N-1}$ and $(w_i)_{i=0..N-1}$, find $(p_i)_{i=0..N-1}$ $(p_i = 0 \text{ or } 1)$ that maximizes $\sum_{i=0}^{N-1} p_i \cdot c_i$ under the condition $\sum_{i=0}^{N-1} p_i \cdot w_i \leq W$. Informally, W, $(c_i)_{i=0..N-1}$ and $(w_i)_{i=0..N-1}$ represent the weight capacity of a given knapsack, the value and weight of each item, respectively, and the problem is to pack items so that the sum of their values shows its maximum. We assume that $c_i > 0$ $(0 \leq i < N)$, $w_i > 0$ $(0 \leq i < N)$, $\frac{c_i}{w_i} \geq \frac{c_j}{w_j}$ $(0 \leq i < j < N)$ for simplicity.

```
1: double max = -1.0, c[N], w[N];
2: int tmp[N], p[N];
3: void knapsack(int i, double sum, double weight)
4: {
5:   double z, relaxedks();
6:   if (i == N) {
7:     if (sum > max) {
8:       max = sum;
9:       for (int j = 0; j < N; j++) p[j] = tmp[j];
10:    }
11:  } else {
12:    if (w[i] <= weight) {
13:      tmp[i] = 1;
14:      knapsack(i+1, sum+c[i], weight-w[i]);
15:    }
16:    z = relaxedks(i+1, weight);
17:    if (z+sum > max) {
18:      tmp[i] = 0;
19:      knapsack(i+1, sum, weight);
20:    }
21:  }
22:}
23:double relaxedks(int i, double weight)
24:{
25:  int j;
26:  double r = 0.0;
27:  for (j = i; j < N; j++) {
28:    if (w[j] < weight) {
29:      r += c[j]; weight -= w[j];
30:    } else return r + weight * c[j]/w[j];
31:  }
32:  return r;
33:}
34:void main()
35:{
36:  knapsack(0,0.0, W);
37:}
```

(1) Let N=5, W=8.0, $(c_i)_{i=0..4}$ = (3.0, 4.0, 7.0, 5.0, 7.0), and $(w_i)_{i=0..4}$ = (1.0, 2.0, 4.0, 3.0, 5.0). Find a solution $(p_i)_{i=0..4}$.

(2) When we execute the function `main` by setting the values shown in Question (1) to `N`, `W`, `c[]` and `w[]`, enumerate the calls of the function `knapsack` with its arguments in the order of calls until termination.

(3) Consider the relaxed condition: a real number $0 \leq p_i \leq 1$ can be taken for $(p_i)_{i=0..N-1}$. Under this condition together with $\sum_{i=0}^{N-1} p_i \cdot w_i \leq W$, find $(p_i)_{i=0..N-1}$ that maximizes $\sum_{i=0}^{N-1} p_i \cdot c_i$. Refer to the function `relaxedks`.

(4) In the if-statement of lines 17—20, the function `knapsack` may be further called only when the condition `(z+sum > max)` is true. Explain why this is allowed in finding a solution.

(5) Explain why the given program actually tests at least one of the solutions, and consequently, gives a solution to the knapsack problem.

(6) If 0.0 is set as the initial value of `max`, there are some cases where the program never executes line 9, which corresponds to assigning a solution to `p[]`. Specify such a case and give its reason.