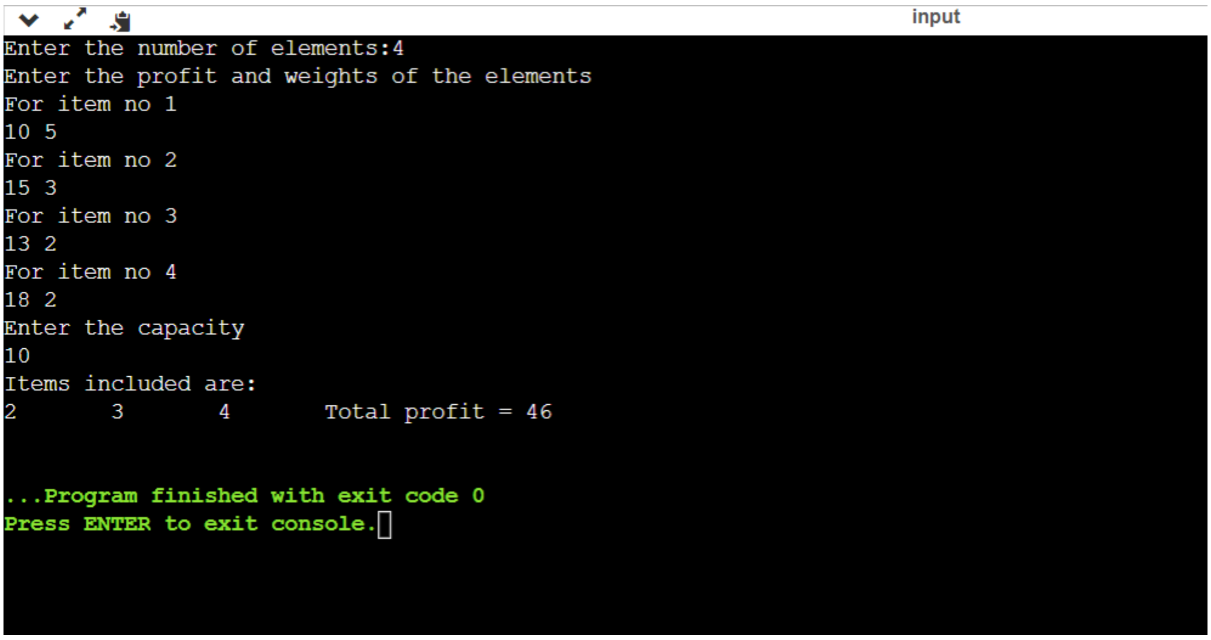
KNAPSACK PROBLEM

```
#include <stdio.h>

#define MAX_ELEMENTS 10

int w[MAX_ELEMENTS], p[MAX_ELEMENTS], v[MAX_ELEMENTS][MAX_ELEMENTS], n,
i, j, cap, x[MAX_ELEMENTS] = {0};

int max(int i, int j) {
   return ((i > j) ? i : j);
}

int knap(int i, int j) {
   int value;
   if (v[i][j] < 0) {
      if (j < w[i])
         value = knap(i - 1, j);
      else
         value = max(knap(i - 1, j), p[i] + knap(i - 1, j - w[i]));
      v[i][j] = value;
   }
   return v[i][j];
}

int main() {
   int profit, count = 0;

   printf("Enter the number of elements:");
   scanf("%d", &n);
   if (n <= 0 || n > MAX_ELEMENTS) {
      printf("Invalid number of elements.\n");
      return 1;
   }

   printf("Enter the profit and weights of the elements\n");
   for (i = 1; i <= n; i++) {
      printf("For item no %d\n", i);
      scanf("%d%d", &p[i], &w[i]);
   }

   printf("Enter the capacity \n");
   scanf("%d", &cap);
   if (cap <= 0) {
      printf("Invalid capacity.\n");
      return 1;
   }

   for (i = 0; i <= n; i++)
```

```
        for (j = 0; j <= cap; j++)
            if ((i == 0) || (j == 0))
                v[i][j] = 0;
            else
                v[i][j] = -1;

    profit = knap(n, cap);
    i = n;
    j = cap;

    while (j != 0 && i != 0) {
        if (v[i][j] != v[i - 1][j]) {
            x[i] = 1;
            j = j - w[i];
            i--;
        } else
            i--;
    }

    printf("Items included are:\n");
    for (i = 1; i <= n; i++)
        if (x[i])
            printf("%d\t", i);

    printf("Total profit = %d\n", profit);

    return 0;
}
```



```
Enter the number of elements:4
Enter the profit and weights of the elements
For item no 1
10 5
For item no 2
15 3
For item no 3
13 2
For item no 4
18 2
Enter the capacity
10
Items included are:
2       3       4       Total profit = 46


...Program finished with exit code 0
Press ENTER to exit console.
```