DAA Lab-6

Name: Kshitij Kumar Sharma Roll No.: 1905514 Date: 20/08/2021

Q-6.1) Packing unbreakable items to maximize profit. Given a set of n items and a container of the capacity C. Additioally, for each item the weight as well as value are also known. Then, the problem is to select a subset of the given items that can be packed in the container to maximize the value such that

1) An item cannot be selected partially (unbreakable).
2) Total weight of the packed items does not exceed the capacity of the container C.

Approach 1)

1) Sort items in the decreasing order of the ratio of value/weight.
2) Consider items one by one and include it if the total weight of packed items does not exceed the capacity C.

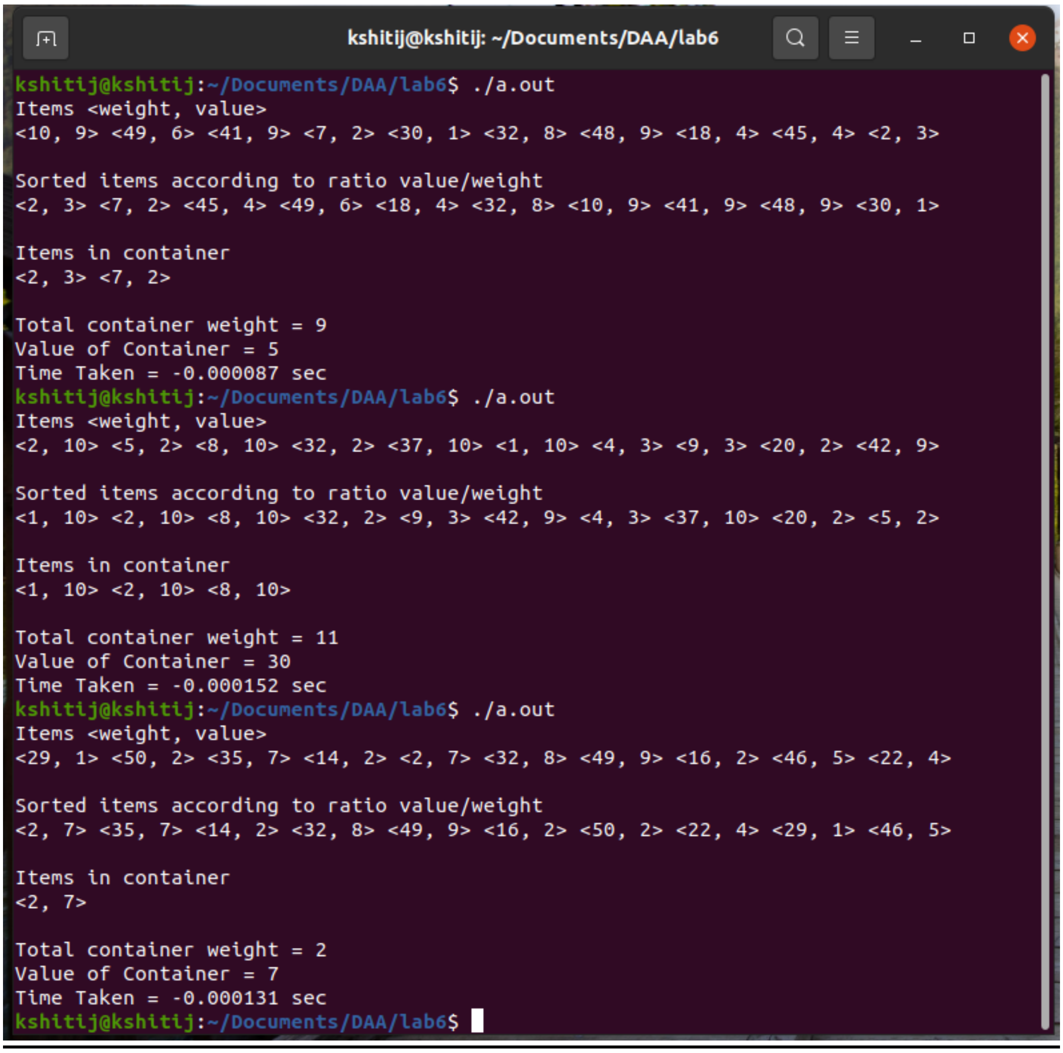
Program:

```
/*
Written by: Kshitij Kumar Sharma                    Roll No.: 1905514
Idea of the solution:
        I have taken item and container structure to keep the details of the items and containers details, I have used quick sort for sorting of the structure with respect to the ratio of value/weight. In this one the capacity of items cannot be divided.
*/
#include<bits/stdc++.h>
#include<stdio.h>
#include<stdlib.h>
#include<time.h>

using namespace std;

const int capacity=20;
int n=10;

struct item                                     //item structure
   {
   int id,weight;
   float value,ratio;
   }s[10],t;                                    //global structures

struct container                                //container
   {
   int id[10] ,weight;
   float value;
   };
/*
float random_float(float a, float b)      // for generating random float value in a range
   {
   float random=((float)rand())/(float)b;
   float diff=b-a;
   float r=random*diff;
   return a+r;
   }
*/
int partition(int p,int r)                    // part of random quick sort algorithm
   {
       int j,i=p-1;
       int pivot=s[r].ratio;

       for(j=p;j<=r-1;j++)
           {
               if(s[j].ratio>=pivot)
                   {
                       i++;
                       t=s[i];
                       s[i]=s[j];
                       s[j]=t;
                   }
           }
        t=s[i+1];
        s[i+1]=s[r];
        s[r]=t;
        return i+1;
   }
int random_partition(int p,int r)                  // part of random quick sort algorithm
   {
       srand(time(0));
       int i=(rand()%(r-p+1))+p;
       t=s[r];
       s[r]=s[i];
       s[i]=t;
       return partition(p,r);
   }
void random_quick_sort(int p,int r)                      //random quick sort.
   {
       int q;
       if(p<r)
        {
               q=random_partition(p,r);
               random_quick_sort(p,q-1);
               random_quick_sort(q+1,r);
        }
   }

int pos(int a)                                     //returns position of given item id
```

```
   {
      int i;
      for(i=0;i<n;i++)
         {
         if(a==s[i].id)
            return i;
         }
   }

void print()
   {
   int i;
   for(i=0;i<n;i++)
      {
       cout<<"<"<<s[i].weight<<", "<<s[i].value<<">"<<" ";
      }
   }
int main()
   {
      int i,j,added_weight=0;
      struct container p;
      clock_t start,end;
      double duration;
      p.weight=0;
      p.value=0;
      srand(time(0));
      for(i=0;i<n;i++)                              //generating random struct details
         {
         s[i].id=i+1;
         s[i].weight=(rand()%(50-1+1))+1;
         s[i].value=(rand()%(10-1+1))+1;            //random_float(0.01,0.3);
         s[i].ratio= s[i].value/s[i].weight;
         }
      start=clock();
      cout<<"Items <weight, value> \n";
      print();
      cout<<endl;
      random_quick_sort(0,n-1);

      cout<<"\nSorted items according to ratio value/weight \n";
      print();
      cout<<endl;
      i=0;
      added_weight=s[0].weight;
      while(added_weight<=capacity)                          //adding items into the container
         {
         p.id[i]=s[i].id;
```

```
        p.weight=p.weight+s[i].weight;
        p.value=p.value+s[i].value;
        i++;
        added_weight=added_weight+s[i].weight;
        }
    cout<<"\nItems in container \n";
    for(j=0;j<i;j++)
        {
         cout<<"<"<<s[pos(p.id[j])].weight<<", "<<s[pos(p.id[j])].value<<">"<<" ";
        }
    cout<<"\n\nTotal container weight = "<<p.weight<<endl;
    cout<<"Value of Container = "<<p.value<<endl;
    end=clock();
    duration=(double(start-end))/CLOCKS_PER_SEC;
    printf("Time Taken = %f sec\n",duration);
  }
```

Output:



```
kshitij@kshitij:~/Documents/DAA/lab6$ ./a.out
Items <weight, value>
<10, 9> <49, 6> <41, 9> <7, 2> <30, 1> <32, 8> <48, 9> <18, 4> <45, 4> <2, 3>

Sorted items according to ratio value/weight
<2, 3> <7, 2> <45, 4> <49, 6> <18, 4> <32, 8> <10, 9> <41, 9> <48, 9> <30, 1>

Items in container
<2, 3> <7, 2>

Total container weight = 9
Value of Container = 5
Time Taken = -0.000087 sec
kshitij@kshitij:~/Documents/DAA/lab6$ ./a.out
Items <weight, value>
<2, 10> <5, 2> <8, 10> <32, 2> <37, 10> <1, 10> <4, 3> <9, 3> <20, 2> <42, 9>

Sorted items according to ratio value/weight
<1, 10> <2, 10> <8, 10> <32, 2> <9, 3> <42, 9> <4, 3> <37, 10> <20, 2> <5, 2>

Items in container
<1, 10> <2, 10> <8, 10>

Total container weight = 11
Value of Container = 30
Time Taken = -0.000152 sec
kshitij@kshitij:~/Documents/DAA/lab6$ ./a.out
Items <weight, value>
<29, 1> <50, 2> <35, 7> <14, 2> <2, 7> <32, 8> <49, 9> <16, 2> <46, 5> <22, 4>

Sorted items according to ratio value/weight
<2, 7> <35, 7> <14, 2> <32, 8> <49, 9> <16, 2> <50, 2> <22, 4> <29, 1> <46, 5>

Items in container
<2, 7>

Total container weight = 2
Value of Container = 7
Time Taken = -0.000131 sec
kshitij@kshitij:~/Documents/DAA/lab6$ 
```

Q-6.2)

Packing breakable items to maximize profit.

Approach 2)

1) Sort items in the decreasing order of the ratio of value/weight.
2) Consider items one by one and include it fully if the total weight of packed items does not exceed the capacity C or the maximum portion of it such that capacity condition is not violated.
3) Return if container is full.

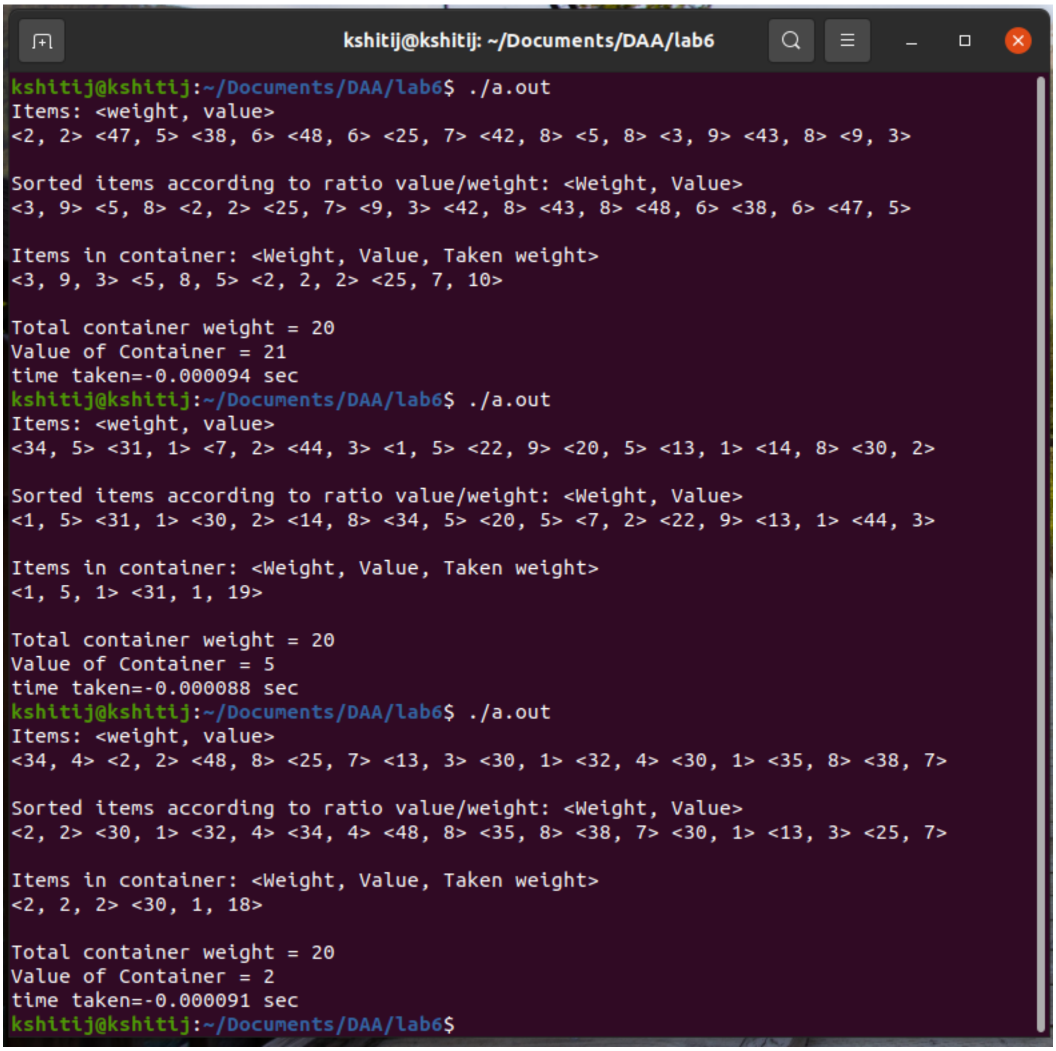
Program:

```
/*
Written by: Kshitij Kumar Sharma                         Roll No.: 1905514
Idea of the solution:
        I have taken item and container structure to keep the details of the items and containers
details, I have used quick sort for sorting of the structure with respect to the ratio of value/weight.
In this one the capacity of items can be divided.
*/
#include<bits/stdc++.h>
#include<stdio.h>
#include<stdlib.h>
#include<time.h>

using namespace std;
                                                         //everything remains same as previous one
const int capacity=20;
int n=10;

struct item
   {
   int id,weight,taken_weight;
   float value,ratio;
   }s[10],t;
struct container
   {
   int id[10] ,weight;
   float value;
   };
/*
float random_float(float a, float b)
   {
    float random=((float)rand())/(float)b;
   float diff=b-a;
   float r=random*diff;
   return a+r;
   }
*/
int partition(int p,int r)
   {
      int j,i=p-1;
      int pivot=s[r].ratio;

      for(j=p;j<=r-1;j++)
         {
            if(s[j].ratio>=pivot)
               {
                  i++;
                  t=s[i];
                  s[i]=s[j];
                  s[j]=t;
               }
         }
         t=s[i+1];
         s[i+1]=s[r];
         s[r]=t;
         return i+1;
   }
int random_partition(int p,int r)
   {
      srand(time(0));
      int i=(rand()%(r-p+1))+p;
      t=s[r];
      s[r]=s[i];
      s[i]=t;
      return partition(p,r);
   }
void random_quick_sort(int p,int r)
   {
      int q;
      if(p<r)
         {
            q=random_partition(p,r);
            random_quick_sort(p,q-1);
            random_quick_sort(q+1,r);
         }
   }

int pos(int a)
```

```
   {
      int i;
      for(i=0;i<n;i++)
         {
         if(a==s[i].id)
            return i;
         }
   }

void print()
   {
   int i;
   for(i=0;i<n;i++)
      {
       cout<<"<"<<s[i].weight<<", "<<s[i].value<<">"<<" ";
      }
   }
int main()
   {
      int i,j,added_weight=0;
      struct container p;
      clock_t start,end;
      double duration;
      p.weight=0;
      p.value=0;
      srand(time(0));
      for(i=0;i<n;i++)
         {
         s[i].id=i+1;
         s[i].weight=(rand()%(50-1+1))+1;
         s[i].value=(rand()%(10-1+1))+1;            //random_float(0.01,0.3);
         s[i].ratio= s[i].value/s[i].weight;
         s[i].taken_weight=0;
         }
      start=clock();
      cout<<"Items: <weight, value> \n";
      print();
      cout<<endl;
      random_quick_sort(0,n-1);
      cout<<"\nSorted items according to ratio value/weight: <Weight, Value> \n";
      print();
      cout<<endl;
      i=0;
      added_weight=s[0].weight;
```

```
    int lw;
    while(added_weight<capacity)
      {
      p.id[i]=s[i].id;
      p.weight=p.weight+s[i].weight;
      p.value=p.value+s[i].value;
      s[i].taken_weight=s[i].weight;
      lw=capacity-added_weight;
      if(lw<s[i+1].weight)                        // breaking the item and adding to container
        {
          p.id[i+1]=s[i+1].id;
          p.weight=p.weight+lw;
          p.value=p.value+int(s[i+1].value*lw/s[i+1].weight);
          s[i+1].taken_weight=lw;
          added_weight=added_weight+lw;
          i++;
          break;
        }
      i++;
      added_weight=added_weight+s[i].weight;
      }
    cout<<"\nItems in container: <Weight, Value, Taken weight> \n";
    for(j=0;j<=i;j++)
      {
       cout<<"<"<<s[pos(p.id[j])].weight<<", "<<s[pos(p.id[j])].value<<", "<<s[pos(p.id[j])].taken_w
eight<<">"<<" ";
      }
    cout<<"\n\nTotal container weight = "<<p.weight<<endl;
    cout<<"Value of Container = "<<p.value<<endl;
    end=clock();
    duration=((double)(start-end)/CLOCKS_PER_SEC);
    printf("time taken=%f sec\n",duration);
  }
```

Output:



```
kshitij@kshitij:~/Documents/DAA/lab6$ ./a.out
Items: <weight, value>
<2, 2> <47, 5> <38, 6> <48, 6> <25, 7> <42, 8> <5, 8> <3, 9> <43, 8> <9, 3>

Sorted items according to ratio value/weight: <Weight, Value>
<3, 9> <5, 8> <2, 2> <25, 7> <9, 3> <42, 8> <43, 8> <48, 6> <38, 6> <47, 5>

Items in container: <Weight, Value, Taken weight>
<3, 9, 3> <5, 8, 5> <2, 2, 2> <25, 7, 10>

Total container weight = 20
Value of Container = 21
time taken=-0.000094 sec
kshitij@kshitij:~/Documents/DAA/lab6$ ./a.out
Items: <weight, value>
<34, 5> <31, 1> <7, 2> <44, 3> <1, 5> <22, 9> <20, 5> <13, 1> <14, 8> <30, 2>

Sorted items according to ratio value/weight: <Weight, Value>
<1, 5> <31, 1> <30, 2> <14, 8> <34, 5> <20, 5> <7, 2> <22, 9> <13, 1> <44, 3>

Items in container: <Weight, Value, Taken weight>
<1, 5, 1> <31, 1, 19>

Total container weight = 20
Value of Container = 5
time taken=-0.000088 sec
kshitij@kshitij:~/Documents/DAA/lab6$ ./a.out
Items: <weight, value>
<34, 4> <2, 2> <48, 8> <25, 7> <13, 3> <30, 1> <32, 4> <30, 1> <35, 8> <38, 7>

Sorted items according to ratio value/weight: <Weight, Value>
<2, 2> <30, 1> <32, 4> <34, 4> <48, 8> <35, 8> <38, 7> <30, 1> <13, 3> <25, 7>

Items in container: <Weight, Value, Taken weight>
<2, 2, 2> <30, 1, 18>

Total container weight = 20
Value of Container = 2
time taken=-0.000091 sec
kshitij@kshitij:~/Documents/DAA/lab6$
```