ADT Bag

- hidden representation
- generic elements

```
// Element.h
#ifndef _ELEMENT_H
#define _ELEMENT_H
   typedef void *TE;
   typedef int (*CmpFunc)(TE, TE);
   typedef void (*DelFunc)(TE &);
   typedef void (*CpyFunc)(TE &, TE);
#endif

// Bag.h
#ifndef _BAG_H
#define _BAG_H
#include "Element.h"
   Struct _Bag;
   Typedef _Bag * Bag;
   void create (Bag &, CmpFunc, CpyFunc, DelFunc);
   void destroy (Bag &);
   int card (Bag);
   void add (Bag, TE);
   Bag reunion (Bag, Bag);
   TE* toArray (Bag);
#endif
```

```
//Bag.cpp
 # include "Bag.h"
 # include <stdio.h>
  structure _Bag {
          TE* el; //array de adrese
          int* f;
          int l;
          int cap;
          CmpFunc cmp;
          delFunc del;
  };      CpyFunc cpy;
```

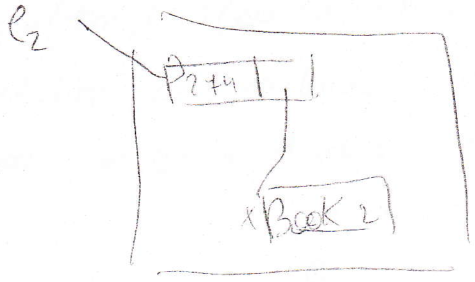
```
Bag b;
 create(b, --)
```

```
void create (Bag &b, CmpFunc cmp, CpyFunc cpy, delFunc del) {
        b=new _Bag;
        b->l =0;
        b->cap=10;
        b->cmp = cmp;
        b->cpy =cpy;
        b->del =del;
        b->el = new TE[b->cap];
        b->f = new int[b->cap];
}

void  destroy (Bag &b) {
     If (b!=NULL)
          If (b->el!=NULL) {
               for (int i=0; i<b->~~cap~~l; i++) {
                    b->del (b->el[i]);
               }
               delete []b->el;
          }
          If (b->f!=NULL)
               delete []b->f;
     }
```

```
    delete b;
    b=NULL;
  }.

void resize (Bag b) {
    TE* new_el;
    b -> cap = b -> cap*2;
    new_el = new TE [b->cap];

    new_f = new int [b->cap];
    for (int i=0 ; i<b ->l ; i++) {
            new_el [i] = b-> el [i];
            new_f[i] = b -> f [i];
    }
    delete [] b->el;
    delete [] b->f;
    b-> el = new_el;
    b-> f = new_f;
}.


void add ( Bag b , TE ~~el~~ elem){
    bool found = false;
    int i=0;
    while (found = false) && (i<b->l)) {
        if (b ->cmp (b-> el[i], elem)==0)
               found = true
        else i++;
    }
    if (found ==true) ~~{~~
        ~~b -> l == b->cap)~~
        b ->f[i] ++;
    else {
     if (b ->l== b ->cap)
             resize(b)
            b->cpy(b->el[b->l], elem);
            b->f[b->l]=1;
            b->l++;
        }
    }

Bag reunion (Bag a, Bag b) {
        Bag c;
        create (c, a->cmp, a->cpy, a->del);
        for (int i=0; i<a->l; i++)                    | copiază din a el + f în c
            for (int j=0; j<a->f[i], j++)             |
                add (c, a->el[i]);                    |

        for (int i=0; i<b->l; i++)                    | copiază din b el+f
            for (int j=0; j<b->f[i]; j++)             |
                add (c, b->el[i]);                    |
                                                      |
        return c                                      |
}

TE* toarray (Bag b) {
    TE* elements;
    elements =new TE [card (b)];
    int k=-1;
    for (int i=0; i<b->l; i++)
        for (int j=0; j<b->f[i]; j++) {
            k++;
            b->cpy (elements[k]; b->el[i]);
        }
    return elements;
}
```

```
int cand (Bag b) {
    int s=0;
    For (int i=0, i<b->l; i++)
        s = s+ b->f[i];
    return s;
}
```

```
// Test.cpp
# include "Bag.h"
# include <fstream>
# include <string.h>
  ~~type~~
  Struct Book {
        char *nume;
        int nrpag;
  };

  int BookCmp (TE* e1, TE* e2) {
      Book *pb1, *pb2;
      pb1=(book*) e1;
      pb2 = (book *) e2;
      return ~~[illegible]~~strcmp(pb1-> nume; pb2-> nume);
  }

                    d        >.
  void BookCpy( TE& e1, TE e2) {
      Book *pb1, *pb2;
      pb2 =(book *)e2;
      pb1 = new Book
      pb1 -> nrpag= pb2 ->nrpag;
      pb1 -> nume = new char[strlen(pb2->nume)+1] ;
      strcpy (pb1->nume, pb2->nume);
  }.
```



--> void =OK.

```
void BookDel (TE & e) {
        Book * pb ;
        pb = (book *) e ;
        if (pb-> name != NULL)
              delete [] pb -> name ;
         delete pb;
        e=NULL;
}.
```

```
#include <iostream>
 using namespace std;
 int addition (int a, int b)
   { return (a+b);}
 int substraction (int a, int b)
   { return (a-b);}

 int operation (int x, int y, int (*fCall)(int,int))
   { int g;
      g= (*fCall)(x,y);
      return(g);
   }
 int main ()
  { int m,n;
    int (*minus)(int,int)=substraction;
    m = operation (7,5, addition);
    n = operation (20, m, minus);
    cout <<n;
    return 0;
  }.
```

```
- - - -
void increase (void* data, int psi
 { if (psize = sizeof(char))
    { char * pchar;
      pchar = (char *) data;
      ++ (*pchar);
    }
  elseif (psize == sizeof (int)
    { int * pint ;
      pint=(int*) data ;
       ++ (*pint); }
 }
 int main ()
  { char a='x';
    int b=1602;
    increase (& a, sizeof(a));
    increase (&b, sizeof(b));
    cout << a << ", " <<b<<e
    return0;
  }
```

=y, 1603.