```
Dark gray:  Sample solution to the given tasks
```

main.sml

```
fun chooser ( xs : string list , c : Store) =
        if String.compare(hd(xs),"quit")=EQUAL andalso List.length(xs)=1 then (false,c)
        else if String.compare(hd(xs),"create")=EQUAL andalso List.length(xs)=4 then (createColl xs c)
        else if String.compare(hd(xs),"insert")=EQUAL andalso List.length(xs)=4 then insert xs c
        else if String.compare(hd(xs),"delete")=EQUAL andalso List.length(xs)=4 then delete xs c
        else (* call the parser *)
            let
                val out=TextIO.print "\n\t\tResult "
                val r=printColl(p( (listToChar (fineTokenize(xs)) ),coll(#"_",#"b",[]),c))
            in
                (true,c)
            end;
```

storage.sml

```
(* delete the given element in the given collection *)
fun deleteElementS3( [] : Collection list) (e : int) (n : char) = [] : Collection list
    |deleteElementS3( coll(cn,cb,cc)::xs ) (e : int) (n : char) =
        if Char.compare(cn,n)=EQUAL then deleteElement(coll(cn,cb,cc),e)::xs
        else coll(cn,cb,cc)::(deleteElementS3 xs e n);

fun deleteElementS2 ( content(ls) : Store) ( e: int) (n: char) =
        content(deleteElementS3 ls e n);

(* s - Store, e - Element to be deleted, n - name of the collection *)
fun deleteElementS( s : Store) (e : string ) (n : string) =
        deleteElementS2 s (stringToInt(e)) (stringToChar(n));

(* Delete a value from a collection in the store *)
fun delete ( x : string list) ( s : Store) =
        if  String.compare( List.nth(x,0),"delete")=EQUAL andalso
            existsCollS(List.nth(x,3),s) andalso
            String.compare(List.nth(x,2),"from")=EQUAL
        then
            let
                val name=List.nth(x,3)
                val element=List.nth(x,1)
            in
                (true, deleteElementS s element name)
            end
        else
            (true,s);
```

collections.sml

```
(* Converts a bag collection to a set collection *)
fun toSet(coll(n,_,l)) = coll(n,#"s",setof l);

(* Converts a set collection to a bag collection *)
fun toBag(coll(n,_,l)) = coll(n,#"b",l);
```

```
(* delete an element from a collection *)
fun deleteElement( coll(n1,b,c), n : int) = coll(n1,b,del(n,c));

(* helper functions *)
fun reduce f nil a = a
|       reduce f (hd::tl) a = f (hd, reduce f tl a)

fun combine f l1 l2 = reduce
        (fn (x,a) => a@fillList(x, f(count(x,l1), count(x,l2))))
        (singleList(l1,l2));
        []


(* bag operations *)
fun bagUnion ( coll(n1,_,l1), coll(n2,_,l2)) =
        coll (#"_",#"b",combine (fn (c1,c2) => max(c1,c2)) l1 l2);

fun bagIntersection( coll(n1,_,l1), coll(n2,_,l2)) =
        coll (#"_",#"b",combine (fn (c1,c2) => min(c1,c2)) l1 l2);

fun bagDifference ( coll(n1,_,l1), coll(n2,_,l2)) =
        coll (#"_",#"b",combine (fn (c1,c2) => if (c1-c2)>0 then c1-c2 else 0) l1 l2);

(* set operations *)
fun setUnion ( coll(n1,_,l1), coll(n2,_,l2)) =
        coll (#"_",#"s",singleList(l1,l2));

fun setIntersection( coll(n1,_,l1), coll(n2,_,l2)) =
        coll (#"_",#"s",combine (fn (1,1) => 1 | (_,_) => 0) l1 l2);

fun setDifference ( coll(n1,_,l1), coll(n2, _,l2)) =
        coll (#"_",#"s",combine (fn (1,0) => 1 | (_,_) => 0) l1 l2);
```