[OOP]Assign6

Student ID: 20152008

Name: Kusdavletov Ernar

Date: 2017.06.01

1. Descriptions

GenericSet is a class template so that any data type can be stored in a set or bag object. It is an abstract class template, since it has two pure virtual functions, Insert() and Delete(). **Set** and **Bag** are two class templates derived from GenericSet. They are can be used with any data types. A set is a collection of objects which must have distinct values, while a bag is a collection of objects which can have equivalent values. In this problem, we assume that a set or a bag can contain 30 objects at maximum.

Insert() and Delete() in Set:
If a set is full and **Insert()** function is called, a C-string exception **"Set Exception: No more space"** is thrown to a caller function. If an item is already in the set, the item is not inserted into the set and no exception is thrown even if the set is already full. If there are spaces in set, each item is inserted into the member array (called values) in insertion order. I.e., the first inserted data item is stored in values[0], the second item in values[1], and so on. **Delete()** function searches and deletes an item provided as the parameter from the set if it is in the set. The insertion order is preserved even if an item is deleted in the middle of the set. If there's no such item in a set, Delete() function is throwing a C-string exception **"Set Exception: Not found"**.

Insert() and Delete() in Bag:
If a bag is full and **Insert()** function is called, a C-string exception **"Bag Exception: No more space"** is thrown to a caller function. If there are spaces in the bag, an item is inserted into the member array in insertion order, regardless of whether the same item is already in the bag or not (unlike Set). **Delete()** function searches and deletes an item provided as the parameter from the bag if it is in the bag. The insertion order is preserved even if an item is deleted in the middle of the bag. If there are multiple items with the parameter value, the last inserted item is deleted. For example, consider a bag = {1, 3, 4, 1, 2}. After deleting item 1, the bag is {1, 3, 4, 2}. If there's no such item in a bag, Delete() function is throwing a C- string exception **"Bag Exception: Not found"**.

2. Code

```
#ifndef __GENERICSET_H__
#define __GENERICSET_H__

#define MAX_ITEMS 30

#include <iostream>
using namespace std;

typedef int ItemType;

class GenericSet{
    protected: //because we need to use them in derived classes
        int size;
        ItemType values[MAX_ITEMS];
    public:
        GenericSet(); // Default Constructor - an empty generic set is created
        virtual void Insert(ItemType item)=0; // Insert function
        virtual void Delete(ItemType item)=0; // Delete function
        void Clear(); // Clear function - all items are removed from the generic set, and the size = 0
        int GetSize(); // GetSize function - returns value is the number of items in the generic set
        bool IsFull(); // IsFull function - returns true if the generic set is full, false otherwise
        bool IsEmpty(); // IsEmpty function - returns true if the generic set is empty, false otherwise
        friend std::ostream& operator<<(std::ostream& out, const GenericSet& s){
            out << "{";
            for (int i = 0; i < s.size; i++){
                out << s.values[i] ;
                if (i < s.size - 1) out << ", ";
            }
            out << "}";
            return out;
        }
};

GenericSet::GenericSet(){ // Default Constructor - an empty generic set is created
  size = 0;
}

void GenericSet::Clear(){ // Clear function - all items are removed from the generic set, the size = 0
  size = 0;
}

int GenericSet::GetSize(){ // GetSize function - returns value is the number of items in the generic set
  return size;
}

bool GenericSet::IsFull(){  // IsFull function - returns true if the generic set is full, false otherwise
  if (size == MAX_ITEMS) // full if size is equal to MAX_ITEMS
    return true;
  else
    return false;
}
```

```
bool GenericSet::IsEmpty(){ // IsEmpty function - returns true if the generic set is empty, false otherwise
   if (size == 0) // empty if size is equal to 0
      return true;
   else
      return false;
}

template <class T> // class template means that functions can be used with any data types
class Set : public GenericSet{ // Set class is derived from GenericSet
   public:
      virtual void Insert(T item){ // overriding Insert() function for Set class
         bool exist = false; // variable to check whether item is in the values[]
         for (int i = 0; i < size; i++){ // checking whether item exists or not
            if (values[i] == item){
               exist = true;
               break;
            }
         }
         if (IsFull() == true && exist == false){ // if item is not in values[] and values[] is full - throw exception
            throw "Set Exception: No more space";
         }
         else if (exist == false){ // if item is not in values[], then add it
            values[size] = item;
            size += 1;
         }
      }
      virtual void Delete(T item){ // overriding Delete() function for Set class
         bool exist = false; // variable to check whether item is in the values[]
         int place; // variable for storing the position of the item in values[]
         for (int i = 0; i < size; i++){ // checking whether item exists or not
            if (values[i] == item){
               exist = true;
               place = i;
               break;
            }
         }
         if (exist == false){ // if item is not in values[] then throw exception
            throw "Set Exception: Not found";
         }
         else {  // else delete the item from values[]
            for (int i = place; i < size - 1; i++){
               values[i] = values[i + 1];
            }
            size -= 1;
         }
      }
};

template <class T> //class template means that functions can be used with any data types
class Bag : public GenericSet{ // Bag class is derived from GenericSet
   public:
      virtual void Insert(T item){ // overriding Insert() function for Bag class
         if (IsFull() == true){ // if values[] is full then throw exception
```

```
            throw "Bag Exception: No more space";
        }
        else{ // else insert item in values[]
            values[size] = item;
            size += 1;
        }
    }
    virtual void Delete(T item){
        bool exist = false; // variable to check whether item is in the values[]
        int place; // variable for storing the position of the item in values[]
        for (int i = size - 1; i >= 0; i--){ // checking whether item exists or not
            if (values[i] == item){
                exist = true;
                place = i; // i is starting from (size - 1) because we need to delete last same value as item
                break;
            }
        }
        if (exist == false){ // if item is not in values[] then throw exception
            throw "Bag Exception: Not found";
        }
        else{ // else delete the item from values[]
            for (int i = place; i < size - 1; i++){
                values[i] = values[i + 1];
            }
            size -= 1;
        }
    }
};

#endif
```

3. Sample input

First sample input

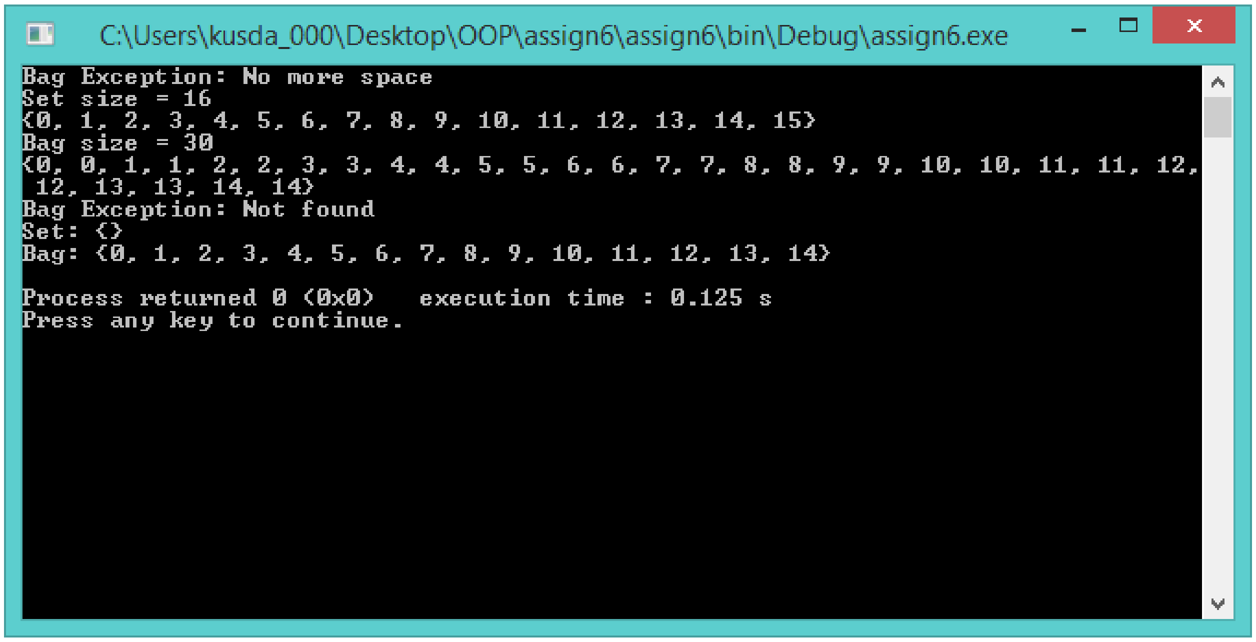
```
int main(){
    Set<int> s1;
    Bag<int> s2;
    try{
        for(int i = 0; i < 31; i++){
            s1.Insert(i / 2);
            s2.Insert(i / 2);
        }
    }
    catch(const char* ex){
        cout << ex << endl;
    }
    cout << "Set size = " << s1.GetSize() << endl;
    cout << s1 << endl;
    cout << "Bag size = " << s2.GetSize() << endl;
    cout << s2 << endl;
    try{
        for(int i=0;i<16;i++){
            s1.Delete(i);
        }
    }
    catch(const char* ex){
        cout << ex << endl;
    }
    try{
        for(int i=0;i<16;i++){
            s2.Delete(i);
        }
    }
    catch(const char* ex){
        cout << ex << endl;
    }

    cout << "Set: " << s1 << endl;
    cout << "Bag: " << s2 << endl;
}
```

Second sample input

```
int main(){
    Set<int> s1;
    Bag<int> s2;
    try{
        for(int i = 0; i < 31; i++){
            s1.Insert(i);
        }
    }
    catch(const char* ex){
        cout << ex << endl;
    }
    cout << "Set size = " << s1.GetSize() << endl;
    cout << s1 << endl;
    cout << "Set is full: " << s1.IsFull() << endl;
    cout << "Set is empty:" << s1.IsEmpty() << endl;
    cout << "Bag size = " << s2.GetSize() << endl;
    cout << s2 << endl;
    cout << "Bag is full: " << s2.IsFull() << endl;
    cout << "Bag is empty:" << s2.IsEmpty() << endl;
    s1.Clear();
    cout << "Set size = " << s1.GetSize() << endl;
    cout << s1 << endl;
    cout << "Set is full: " << s1.IsFull() << endl;
    cout << "Set is empty:" << s1.IsEmpty() << endl;
}
```

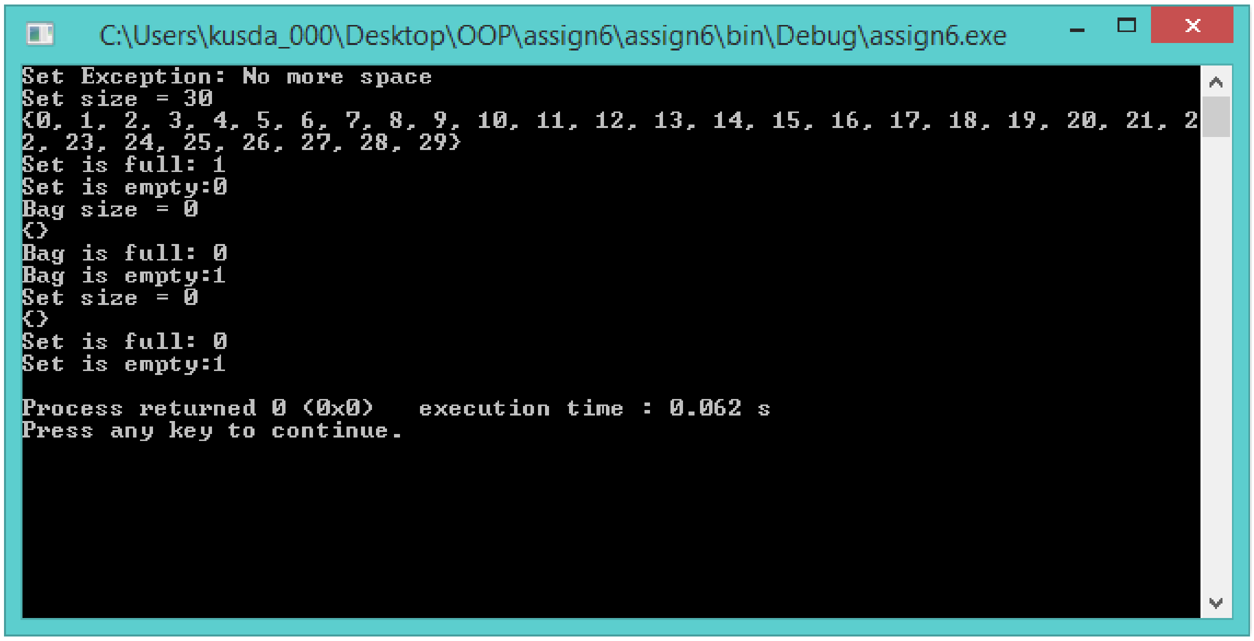
4. Sample output

First sample output


C:\Users\kusda_000\Desktop\OOP\assign6\assign6\bin\Debug\assign6.exe

```
Bag Exception: No more space
Set size = 16
{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15}
Bag size = 30
{0, 0, 1, 1, 2, 2, 3, 3, 4, 4, 5, 5, 6, 6, 7, 7, 8, 8, 9, 9, 10, 10, 11, 11, 12,
 12, 13, 13, 14, 14}
Bag Exception: Not found
Set: {}
Bag: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14}

Process returned 0 (0x0)   execution time : 0.125 s
Press any key to continue.
```

Second sample output


C:\Users\kusda_000\Desktop\OOP\assign6\assign6\bin\Debug\assign6.exe

```
Set Exception: No more space
Set size = 30
{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 2
2, 23, 24, 25, 26, 27, 28, 29}
Set is full: 1
Set is empty:0
Bag size = 0
{}
Bag is full: 0
Bag is empty:1
Set size = 0
{}
Set is full: 0
Set is empty:1

Process returned 0 (0x0)   execution time : 0.062 s
Press any key to continue.
```