10. Übung am 10.01.2011

Anleitung:

Während jeder Übung ist von jeder Gruppe ein (kurzes) Protokoll zu erstellen. Das Protokoll ist eine einfache ASCII-Text-Datei, die mit einem Text-Editor (nedit, kedit, kate) mit dem Sie auch Ihre Programme schreiben, erzeugt wird. Nennen Sie diese Datei unbedingt "PROTOKOLL.txt".

Das Protokoll muss Folgendes enthalten:

1. Datum, Übungsnummer, Gruppennummer, Name(n) der mitwirkenden Studierenden,
2. Benötigter Zeitaufwand für die gestellten Aufgaben (circa),
3. Namen der erstellten Programme (KEINE Listings),
4. Kurze Antwort auf eventuell weiter unten gestellte Fragen,
5. Eventuelle Probleme oder Besonderheiten, falls diese aufgetreten sind.

Sämtliche während der Übung erstellten Dateien (Protokoll, Source Codes, ausführbare Programme, etc.) verbleiben im Verzeichnis für den jeweiligen Übungstag also z.B. "10Ue2011-01-10/" Ihrer Gruppe.

Das Protokoll und die Übungsprogramme sollten am jeweiligen Übungsnachmittag erstellt werden, spätestens jedoch bis zum nächstfolgenden Übungstag (Montag), 14:00! (Spätere Ausarbeitungen können nur in begründeten Fällen und nach Rücksprache mit Ihrem Betreuer bzw. Tutor berücksichtigt werden!)

Aufgabe zu Kapitel 7.2 (0.5 Punkte):

Schreiben Sie unter Verwendung von C-Strings eine Funktion ***char* strDuplicate(const char* s)***, die einen String ***s*** dupliziert. Der neue Speicherplatz soll mit ***new*** angelegt und ***s*** dahin kopiert wird. Ein Zeiger auf den Beginn des Duplikats soll zurückgegeben werden.

```
int main() {
const char * const original =
"Manchmal ist die Kopie besser als das Original!";
 cout << original << endl;
.........
 cout << Kopie << endl;
.........
}
```

Aufgabe zu Kapitel 8 (2.5 Punkte):

Schreiben Sie eine Klasse "intSet", bestehend aus den zwei Dateien "IntSet.h" und "IntSet.cpp", sowie ein Testprogramm "main.cpp", welche die mathematische Menge für ganze Zahlen nachbilden sollen. Es sollen nur die folgenden einfachen Funktionen möglich sein:

(a) void addMember(int el):	Element "el" hinzufügen; falls es existiert nichts tun
(b) void delMember(int el):	Element "el" entfernen; falls es nicht existiert nichts tun
(c) bool isMember(int el):	Gibt an ob Element enthalten
(d) size_t size():	Gibt Anzahl der Element zurück
(e) void show():	Gibt alle Elemente auf Standardanzeige aus
(f) void purgeAll():	Löscht alle Elemente
(g) void getMax():	Gibt größtes Element zurück (Falls die Menge leer ist, d.h. Anzahl der Elemente null ist, soll das Programm mit einer Fehlermeldung abbrechen. Verwenden Sie dafür die vom System zur Verfügung gestellte Funktion ***exit()***)
(h) void getMin():	Gibt kleinstes Element zurück(Falls die Menge leer ist, d.h. Anzahl der Elemente null ist, soll das Programm mit einer Fehlermeldung abbrechen. Verwenden Sie dafür die vom System zur Verfügung gestellte Funktion ***exit()***)

Benutzen Sie zum Speichern der Werte ein "vector<int>" Objekt !!
Das Testprogramm könnte auszugsweise in etwa so aussehen:

```
IntSet mySet;
mySet.addMember(2);   // OK
mySet.addMember (-9); // OK
mySet.addMember (2);  // keine Wirkung, 2 ist schon vorhanden
mySet.delMember(99);  // keine Wirkung, 99 ist nicht vorhanden
mySet.delMember(-9);  // OK
mySet.show();
mySet.purgeAll();
for (int i=17; i < 33, ++i) {
mySet.addMember(i*i);
```

```
}
cout << "Anzahl=" << mySet.size()
     << "   Minimum=" << mySet.getMin() ;
if (mySet.isMember(-11)  {
// usw.........
```