UNIVERSITY of GUELPH

# CIS2520 Data Structures

Fall 2011, Assignment 1

Consider the following files:

**StudentType.h**

```
#define MAXNAMESIZE 20
typedef struct {
    char name[MAXNAMESIZE];
    int grade;
} Student;
```

**StudentInterface.h**

```
#include "StudentType.h"

extern void InitializeStudent (char *name, int grade, Student *S);
extern char *NameOfStudent (Student S);
extern int GradeOfStudent (Student S);
extern void FreeStudent (Student *S);
```

**StudentImplementation.c**

```
#include "StudentInterface.h"

void InitializeStudent (char *name, int grade, Student *S) {
    /* your code here */
}
char *NameOfStudent (Student S) {
    /* your code here */
}
int GradeOfStudent (Student S) {
    /* your code here */
}
void FreeStudent (Student *S) {
    /* your code here */
}
```

**ListType.h**

```c
#include "StudentInterface.h"
typedef Student Item;

#define MAXLISTSIZE 4
typedef struct {
    Item items[MAXLISTSIZE];
    int count;
} List;
```

**ListInterface.h**

```c
#include "ListType.h"

extern void Initialize (List *L);
extern void Insert (Item X, int position, List *L);
extern void Delete (int position, List *L);
extern int Full (List *L);
extern int Empty (List *L);
extern int Length (List *L);
extern void Peek (int position, List *L, Item *X);
extern void Destroy (List *L);
```

**ListImplementation.c**

```c
#include "ListInterface.h"

void Initialize (List *L) {
    /* your code here */
}
void Insert (Item X, int position, List *L) {
    /* your code here */
}
void Delete (int position, List *L) {
    /* your code here */
}
int Full (List *L) {
    /* your code here */
}
int Empty (List *L) {
    /* your code here */
}
int Length (List *L) {
    /* your code here */
}
```

```
void Peek (int position, List *L, Item *X) {
    /* your code here */
}
void Destroy (List *L) {
    /* your code here */
}
```

**myProgram.c**

```
#include "ListInterface.h"

/* your code here */
```

**QUESTION 1.** Add file and function header comments to **StudentInterface.h** and **ListInterface.h**. These comments should be formatted as indicated below:

```
/**********************************************************************
* FILE NAME:
* PURPOSE:
* AUTHOR:
* DATE:
* NOTES:
**********************************************************************/

/**********************************************************************
* FUNCTION NAME:
* PURPOSE:
* ARGUMENTS:
* RETURNS:
* REQUIRES: (preconditions)
* ENSURES: (postconditions)
* NOTES:
**********************************************************************/
```

**QUESTION 2.** Complete the files **StudentImplementation.c** and **ListImplementation.c** where indicated. Include instructions to test the operations' preconditions and post-conditions. These instructions should be compiled only when in debug mode, using the `#define`, `#ifdef` and `#endif` preprocessor directives.

**QUESTION 3.** Complete the file **myProgram.c** where indicated.
**myProgram.c** is a simple test program, which uses the functions declared in **StudentInterface.h** and **ListInterface.h** to create and modify a list of students. The output of the program should be the following:

```
List is empty; list is not full; list is of length 0:
List is not empty; list is not full; list is of length 1:
     John  75%
List is not empty; list is not full; list is of length 2:
     Mary  80%
     John  75%
List is not empty; list is not full; list is of length 3:
     Mary  80%
     John  75%
     Pete  90%
List is not empty; list is full; list is of length 4:
     Mary  80%
     John  75%
     Liz   85%
     Pete  90%
List is not empty; list is not full; list is of length 3:
     John  75%
     Liz   85%
     Pete  90%
List is not empty; list is not full; list is of length 2:
     John  75%
     Pete  90%
List is not empty; list is not full; list is of length 1:
     John  75%
List is empty; list is not full; list is of length 0:
```

**QUESTION 4.** Modify **StudentType.h** as shown below, and modify **Student-Implementation.c** accordingly. Modify **ListType.h** and **ListImplementation.c** to accommodate a one-way linked list representation instead of a sequential list representation. The other files should not be modified, and the output of your test program should be exactly the same.

**StudentType.h**

```
typedef struct {
    char *name;
    int grade;
} Student;
```