UNIVERSITY of GUELPH

CIS2520 Data Structures
Fall 2011, Assignment 2

Download **assign1key.zip**. It will be posted on **Moodle** Oct 5 at 00.05am.
assign1key.zip packs two folders: **List_Student_S** and **List_Student_L**.

QUESTION 1: *Better lists of students (and better students)*

*This question concerns the files in the folder **List_Student_L**.*

a) In **myProgram.c**, replace

```
#include "ListInterface.h"
```

with

```
#include "StudentInterface.h"
#include "ListInterface.h"
```

Including both header files causes a compilation error, because **ListInterface.h** already includes **StudentInterface.h** (through **ListType.h**), and the type **Student** is therefore defined twice (which is not allowed in C). A way to go around this is to use *#include guards*: add appropriate **#ifndef**, **#define** and **#endif** directives in **StudentInterface.h** so that **myProgram.c** compiles.

b) Modify **StudentImplementation.c** and **ListImplementation.c** so that the pre- and post- conditions are checked when in debug mode. Do not use the **#define** directive, and do not use the **printf()** and **exit()** functions. Use the **assert()** macro instead, and modify the **makefile** so that

```
make –B
```

unconditionally makes all targets with debugging ON
(the pre- and post- conditions are checked), while

```
make –B FLAG=-DNDEBUG
```

unconditionally makes all targets with debugging OFF
(the pre- and post- conditions are not checked).

c) Add the lines below to **ListInterface.h**, and implement the function **Reverse()** in **ListImplementation.c** using recursion.

```
/**********************************************************************
 * FUNCTION NAME: Reverse
 * PURPOSE: Reverses a List
 *          (the first Item becomes the last and vice versa).
 * ARGUMENTS: The address of the List to be reversed (List *L)
 **********************************************************************/
extern void Reverse (List *L);
```

QUESTION 2: *From lists of students to stacks of integers*

Make a copy ***Stack_int_L*** *of the revised folder* ***List_Student_L****.*
This question concerns the files in ***Stack_int_L****.*

a) Delete the files **StudentType.h**, **StudentInterface.h** and **StudentImplementation.c**.

b) In **ListType.h**, replace

```
#include "StudentInterface.h"
typedef Student Item;
#define MAXLISTSIZE 4
```

with

```
typedef int Item;
```

c) In **ListInterface.h** , replace

```
#include "ListType.h"
```

with

```
#include "StackType.h"
```

and replace the function declarations with

```
extern void Initialize (Stack *S);
extern void Push (Item X, Stack *S);
extern void Pop (Stack *S);
extern int Full (Stack *S);
extern int Empty (Stack *S);
extern int Length (Stack *S);
extern void Top (Stack *S, Item *X);
extern void Destroy (Stack *S);
```

d) Rename **ListType.h**, **ListInterface.h** and **ListImplementation.c**:
call them **StackType.h**, **StackInterface.h** and **StackImplementation.c**.

e) Replace **test.txt** with:

test.txt

```
6376120394793984100199839835938398392921012673849501
4522801620563928374090928137230475860
```

f) Modify all the files according to the changes above, and so that the program (**a.out**) outputs the sum of the two numbers in **test.txt**. This sum should be calculated using three stacks, as shown in class.

QUESTION 3: *From lists to queues*

*Create a copy **Queue_Student_S** of the folder **List_Student_S**.*
*This question concerns the files in **Queue_Student_S**.*

a) In **ListInterface.h** , replace

```
#include "ListType.h"
```

with

```
#include "QueueType.h"
```

and replace the function declarations with

```
extern void Initialize (Queue *Q);
extern void Enqueue (Item X, Queue *Q);
extern void Dequeue (Queue *Q);
extern int Full (Queue *Q);
extern int Empty (Queue *Q);
extern int Length (Queue *Q);
extern void Head (Queue *Q, Item *X);
extern void Destroy (Queue *Q);
```

b) Rename **ListType.h**, **ListInterface.h** and **ListImplementation.c**:
call them **QueueType.h**, **QueueInterface.h** and **QueueImplementation.c**.

c) Modify all the files according to the changes described in **a)** and **b)**,
and so that the output of the program (**a.out**) is as shown below.
Queues should be implemented using circular arrays.

```
Queue is empty; queue is not full; queue is of length 0:
Queue is not empty; queue is not full; queue is of length 1:
     John  75%
Queue is not empty; queue is not full; queue is of length 2:
     John  75%
     Mary  80%
```

```
Queue is not empty; queue is not full; queue is of length 1:
     Mary  80%
Queue is not empty; queue is not full; queue is of length 2:
     Mary  80%
     Pete  90%
Queue is not empty; queue is not full; queue is of length 3:
     Mary  80%
     Pete  90%
     Liz   85%
Queue is not empty; queue is full; queue is of length 4:
     Mary  80%
     Pete  90%
     Liz   85%
     Bob   60%
Queue is not empty; queue is not full; queue is of length 3:
     Pete  90%
     Liz   85%
     Bob   60%
Queue is not empty; queue is not full; queue is of length 2:
     Liz   85%
     Bob   60%
Queue is not empty; queue is not full; queue is of length 1:
     Bob   60%
Queue is empty; queue is not full; queue is of length 0:
```

SUBMISSION

Make sure the revised folders **List_Student_L**, **Stack_int_L** and **Queue_Student_S** contain text files only (.h, .c, makefile, test.txt). Make sure all the file and function header comments have been updated according to the requested changes.

Place the three folders in a root folder **CIS2520_LastNameFirstName_A2**.
Zip the root folder and upload it to **Moodle** by Oct 16, 11:55pm.

MARKING SCHEME

QUESTION 1 = 30%
QUESTION 2 = 40%
QUESTION 3 = 30%