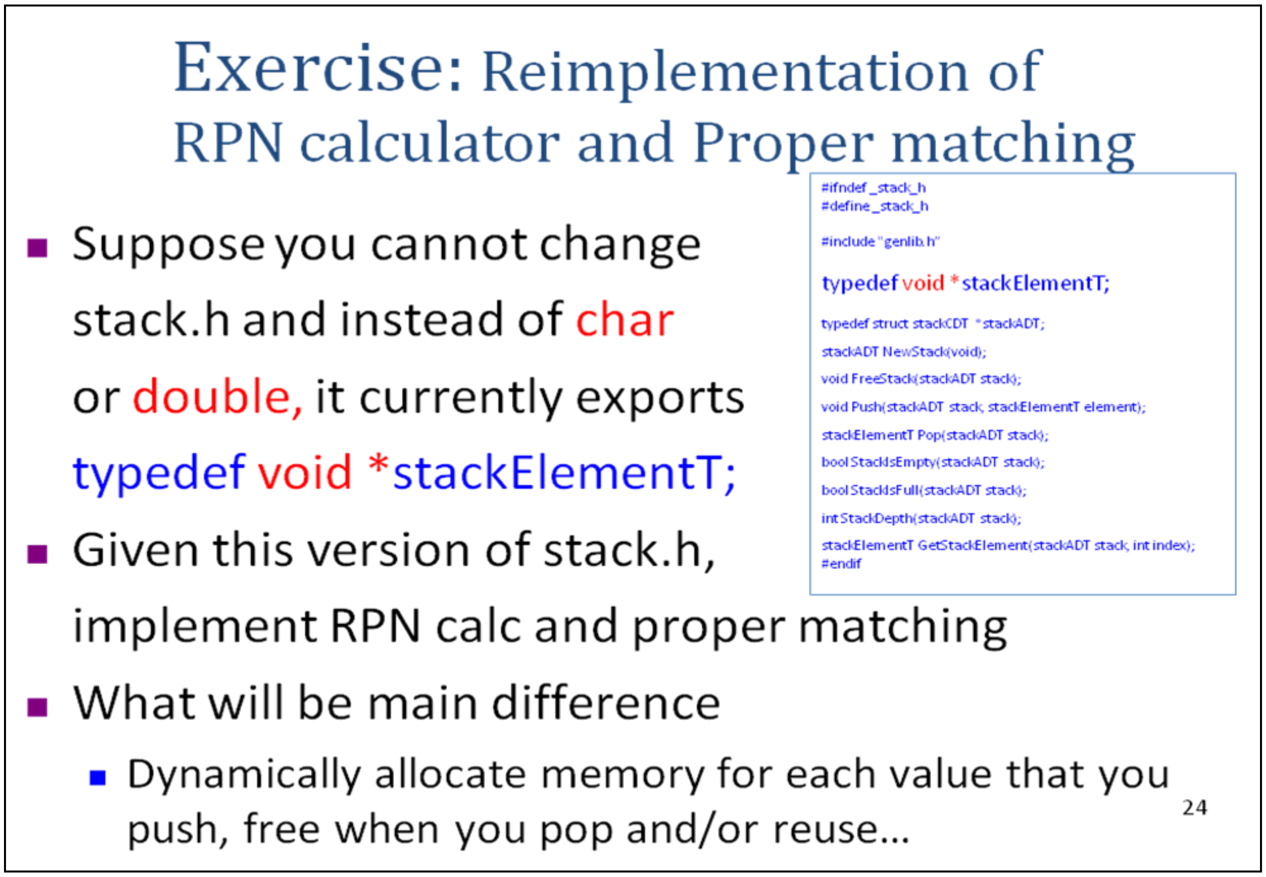
CS 2123 Data Structures Recitation - Exercise 07

(Library – Abstract Data Type)



Exercise: Reimplementation of RPN calculator and Proper matching

- Suppose you cannot change stack.h and instead of char or double, it currently exports typedef void *stackElementT;
- Given this version of stack.h, implement RPN calc and proper matching
- What will be main difference
  - Dynamically allocate memory for each value that you push, free when you pop and/or reuse…

```
#ifndef _stack_h
#define _stack_h

#include "genlib.h"

typedef void *stackElementT;

typedef struct stackCDT *stackADT;

stackADT NewStack(void);

void FreeStack(stackADT stack);

void Push(stackADT stack, stackElementT element);

stackElementT Pop(stackADT stack);

bool StackIsEmpty(stackADT stack);

bool StackIsFull(stackADT stack);

int StackDepth(stackADT stack);

stackElementT GetStackElement(stackADT stack, int index);
#endif
```

24

In this exercise, you are simply asked **to re-implement RPN calculator** by using stack.h that exports

```
      typedef void *stackElementT;
```

(just doind rpn calc is enough to submit, you can do the proper matching on your own time)

First, go to class web site and download the existing stack lib, rpncalc.c and also get books lib.

For this, follow "programs from the textbook" link under online materials in the class webpage
http://www.cs.utsa.edu/~korkmaz/teaching/cs2123/

get `booklib.zip` and `08-Abstract-Data-Types.zip`
save both under a common directory, say X
Then follow these instructions:

```
~/X> unzip booklib.zip
~/X> cd booklib
~/X/booklib> make
```

```
~/X/booklib> cd ..
~/X>

~/X> unzip 08-Abstract-Data-Types.zip
~/X> cd 08-Abstract-Data-Types
~/X/08-Abstract-Data-Types> make
~/X/08-Abstract-Data-Types>
```

Now you can run existing rpncalc program...

But this existing implementation assumes that stack.h is exporting

```
        typedef double stackElementT;
```

So the existing rpncalc.c directly push/pop double values....

But you are required to re-implement it by assuming that stack.h is exporting

```
        typedef void *stackElementT;
```

So, edit stack.h and simply change

```
      typedef double stackElementT;
                  TO
      typedef void *stackElementT;
```

Finally, re-implement (modify) rpncalc.c such that you will dynamically allocate memory for double values and push/pop their addresses into the stack.

As always, make sure you release (free) the dynamically allocated memories if you allocate any memory in your programs. So, before submitting your program, run it with `valgrind` to see if there is any memory leakage…

Also if you need to debug your program, compile your programs with –g option and then run it with `gdb` and/or `ddd`.

What to return: !!!! NO LATE RECITATION ASSIGNMNET WILL BE ACCEPTED !!!

1. Create a directory called LASTNAME_Recitation07 and do all your work under that directory
2. First implement your program as described above...
3. Then compile and run it. Copy/paste the result in an output file, say out07.txt.
3. Finally zip your LASTNAME_Recitation07 directory as a single file LASNAME_Recitation07.zip and Go to BB Learn to submit it as **attachment** before the deadline.

/* Don’t forget to include comments about the problem, yourself and each major step in your program! */

You must submit your work using Blackboard Learn and respect the following rules:

1) All assignments must be submitted as either a zip or tar archive file unless it is a single pdf file.
2) Assignments must include all source code.
3) Assignments must include an output.txt file which demonstrates the final test output run by the student.

If your assignment does not run/compile, the output.txt file should include an explanation of what was accomplished, what the error message was that prevented the student from finishing the assignment and what the student BELIEVES to be the underlying cause of the error.
