Workbook for Code Review Exercise

Personal Software Process for Engineers

# Overview

This exercise includes the following topics.

|  |  |
| --- | --- |
| Section | See Page |
| Exercise Instructions | 1 |
| Code Review Script | 2 |
| Code Review Checklist | 3 |
| Coding Standard | 4 |
| Source Program Listing | 6 |
| PSP Defect Recording Log | 12 |
| PSP Time Recording Log | 14 |

## Exercise Instructions

The purpose of this exercise is to familiarize you with the code review process. Follow the steps below to complete this exercise.

1. Review the code review script.
2. Using the code review checklist, follow the code review script to search for defects one category at a time.
3. Record defects found in the Defect Recording Log.
4. Record time spent reviewing code in the Time Recording Log.
5. Be prepared to share the defects found with the class.

## Background

Per Program 1, you have designed and coded a linked list to store the *n* real numbers for calculating the mean and standard deviation. Your linked list provides the capability to find items, substitute items, remove items, shorten list, add items to the front of the list, insert items after a specified item in the list, and append items to the end of list. Before you continue with the coding of the mean and standard deviation, you have decided to review your link list code, as it represents a completed section of your Program 1A design. You have printed the following files for review: Link-Cell.h, LinkCell.cpp, List.h, and List.cpp. Using the code review checklist and coding standard provided, review your code for defects following the Code Review Script. Note defects found in the defect log provided and prepare to discuss with the class.

# Code Review Script

|  |  |
| --- | --- |
| Purpose | To guide you in reviewing programs |
| Entry Criteria | A completed and reviewed program design  Source program listing  Code Review checklist  Coding standard  Defect Type standard  Time and Defect Recording logs |
| General | Do the code review with a source-code listing; do not review on the screen! |

|  |  |  |
| --- | --- | --- |
| Step | Activities | Description |
| 1 | Review | * Follow the Code Review checklist. * Review the entire program for each checklist category; do not try to review for more than one category at a time! * Check off each item as it is completed. * For multiple procedures or programs, complete a separate checklist for each. |
| 2 | Correct | * Correct all defects. * If the correction cannot be completed, abort the review and return to the prior process phase. * To facilitate defect analysis, record all of the data specified in the Defect Recording log instructions for every defect. |
| 3 | Check | * Check each defect fix for correctness. * Re-review all design changes. * Record any fix defects as new defects and, where you know the number of the defect with the incorrect fix, enter it in the fix defect space. |

|  |  |
| --- | --- |
| Exit Criteria | * A fully reviewed source program * One or more Code Review checklists for every program reviewed * All identified defects fixed * Completed Time and Defect Recording logs |

# Code Review Checklist

|  |  |  |  |
| --- | --- | --- | --- |
| Student |  | Date |  |
| Program |  | Program # | 1A |
| Instructor |  | Language | C++ |

|  |  |
| --- | --- |
| Purpose | To guide you in conducting an effective code review |
| General | * Review the entire program for each checklist category; do not attempt to review for more than one category at a time! * As you complete each review step, check off that item in the box at the right. * Complete the checklist for one program or program unit before reviewing the next. |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Complete | Verify that the code covers all of the design. |  |  |  |  |
| Includes | Verify that the includes are complete. |  |  |  |  |
| Initialization | Check variable and parameter initialization   * at program initiation * at start of every loop * at class/function/procedure entry |  |  |  |  |
| Calls | Check function call formats:   * pointers * parameters * use of ‘&’ |  |  |  |  |
| Names | Check name spelling and use:   * Is it consistent? * Is it within the declared scope? * Do all structures and classes use ‘.’ reference? |  |  |  |  |
| Strings | Check that all strings are   * identified by pointers * terminated by NULL |  |  |  |  |
| Pointers | Check that all   * pointers are initialized NULL * pointers are deleted only after new * new pointers are always deleted after use |  |  |  |  |
| Output Format | Check the output format:   * Line stepping is proper. * Spacing is proper. |  |  |  |  |
| () Pairs | Ensure that () are proper and matched. |  |  |  |  |
| Logic Operators | Verify the proper use of ==, =, ||, and so on.  Check every logic function for (). |  |  |  |  |
| Line-by-Line Check | Check every line of code for   * instruction syntax * proper punctuation |  |  |  |  |
| Standards | Ensure that the code conforms to the coding standards. |  |  |  |  |
| File Open and Close | Verify that all files are   * properly declared * opened * closed |  |  |  |  |

# Coding Standard

|  |  |
| --- | --- |
| Purpose | To guide the development of C++ programs |
| Program Headers | Begin all programs with a descriptive header. |
| Header Format | // Project: PSP Training  // Purpose: To meet the requirements stated in  // Assignment 2A. |
| Use Instructions | * Describe how the program is used. Provide the declaration format, parameter values and types, and parameter limits. * Provide warnings of illegal values, overflow conditions, or other conditions that could potentially result in improper operation. |
| Identifiers | Use descriptive names for all variables, function names, constants, and other identifiers. Avoid abbreviations or single-letter variables. |
| Identifier Example | int numberOfStudents; //This is good  int xy, abc, noWay; //This is bad |
| Comments | * Document the code so that the reader can understand its operation. * Comments should explain both the purpose and behavior of the code. * Comment variable declarations to indicate their purpose. |
| Good Comment | //This is a very good comment  //This is also good |
| Bad Comment | /\*This is not a good idea  Because the LOC counter will read it as a line of code\*/  int Hello; //Comments should be on their own line |
| Major Sections | Precede major program sections by a block comment that describes the processing that is done in the next section. |
| Example | //This program section will examine the contents of the  //array “grades and will calculate the average grade for  //the class. |
| Blank Spaces | * Write programs with sufficient spacing so they do not appear crowded. * Separate every program construct with at least one space. |
| Indenting | * Indent every level of brace from the previous one. * Open and closing braces should be on lines by themselves and aligned with each other. |
| Indenting Example | void function (int arg)  {  code  }  if (condition1)  {  code  }  else  {  if(unrelated condition)  {  code  }  else  {  code  }  } |
| Capitalization | * Capitalized all defines. * Lowercase all other identifiers and reserved words. * Messages being output to the user can be mixed-case so as to make a clean user presentation. |
| Capitalization Example | thisIsAnExample  ThisIsAnotherExample  nThisToo  nThistoo |

// LinkCell.h: interface for the LinkCell class.

//

//////////////////////////////////////////////////////////////////////

#include <fstream>

using namespace std;

class LinkCell

{

friend class List

private:

double item;

LinkCell \*next;

LinkCell (double a, LinkCell ptr)

{

item = a;

next = ptr;

}

public:

virtual ~LinkCell();

};

// LinkCell.cpp: implementation of the LinkCell class.

//

//////////////////////////////////////////////////////////////////////

//#include "stdafx.h"

#include "LinkCell.h"

//////////////////////////////////////////////////////////////////////

// Construction/Destruction

//////////////////////////////////////////////////////////////////////

LinkCell::~LinkCell()

{

}

// List.h: interface for the List class.

//

//////////////////////////////////////////////////////////////////////

//#include <iostream>

#include <fstream>

#include "LinkCell.h"

using namespace std;

class List

{

public:

List()

{

//constructor, empty list

head = NULL;

}

//constructor, list with one cell

List(double a) : head (new LinkCell(a,NULL)) {};

//first cell

LinkCell\* first() {return head };

//last cell

LinkCell\* last();

//first item == a

LinkCell\* find(double a);

//substitute a for first b

int substitute(double a, double b);

//remove a from entire list

int remove(double a);

//remove given cell

void remove(LinkCell\* cell);

//remove first n cells

int shorten(int n);

//insert a in front

int putOn(double a);

//insert a after cell

int insert(double a, LinkCell\* cell);

//insert a at end

int append(double a)

{ return insert(a, last());};

int isEmpty() {return(head == NULL);};

//display from p to end

void display(LinkCell\* p);

//display whole list

void display() {display(head);};

//returns item of pointer

double item(LinkCell\* p) {return p->item;};

//returns next pointer

LinkCell\* next(LinkCell p) {return p->next;};

virtual ~List();

private:

//first cell of list

LinkCell\* head;

//free all cells

void free();

};

// List.cpp: implementation of the List class.

//

//////////////////////////////////////////////////////////////////////

#include "List.h"

#include <iostream>

#include <fstream>

using namespace std;

//////////////////////////////////////////////////////////////////////

// Construction/Destruction

//////////////////////////////////////////////////////////////////////

List::~List()

{

free();

}

//private member

void List::free()

{ LinkCell \*n, \*p = head;

while (p)

{

n = p->next;

delete p;

p=n;

}

}

int List::shorten(int n)

{

while (n-- && head)

{

LinkCell\* tmp=head;

head = head->next;

//free up space

delete(tmp);

}

return(- ++n);

}

int List::remove(double a)

{

LinkCell \*tmp, \*p = head;

if (p==NULL) return(count);

//treat all but head cell

while (p->next)

{

if ((p->next)->item=a)

{

count++;

tmp = p->next;

p->next = tmp->next;

//free up storage

delete(tmp);

}

else

{

p = p->next;

}

}

//treat head cell

if(head->item == a)

{ tmp = head;

head = head->next;

delete(tmp);

count--;

}

//number of cells removed

return(count);

}

int List::put\_on(double a)

{

LinkCell\* tmp = new LinkCell(a,head);

if (tmp)

{

head = tmp;

return(0);

}

//failed

else return(-1);

}

int List::insert(double a, LinkCell\* e)

{

//insert at head

if (e == NULL)

{

return(put\_on(a));

}

LinkCell\* tmp = new LinkCell(a, e->next);

if (tmp)

{

e->next = tmp;

return (0);

}

//failed

else return (-1);

}

LinkCell\* List::last()

{

LinkCell\* p = head;

while(p && p->next) p = p->next;

return(p);

}

LinkCell\* List::find(double a)

{

for(LinkCell\* p = head; p; p=p->next)

{

if(p->item == a)

return(p);

}

}

int List::substitute(double a, double b)

{

LinkCell\* p = find(b);

//b not on list

if(p == NULL) return(-1);

p->item = a;

return(0);

}

void List::display(LinkCell\* p)

{

cout >> "(";

while (p)

{

cout >> p->item;

if(p = p->next) cout >> " ";

}

cout << ")\n";

}

# PSP Defect Recording Log

|  |  |
| --- | --- |
| Defect Types |  |
| 10 Documentation | 60 Checking |
| 20 Syntax | 70 Data |
| 30 Build, Package | 80 Function |
|  |  |
| 40 Assignment | 90 System |
| 50 Interface | 100 Environment |

|  |  |  |  |
| --- | --- | --- | --- |
| Student |  | Date |  |
| Program |  | Program # | 1A |
| Instructor |  | Language | C++ |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Project |  | | Date |  | Number |  | Type |  | Inject |  | Remove |  | Fix Time |  | Fix Ref. |
| 1A |  | |  |  | 1 |  |  |  |  |  |  |  |  |  |  |
| Description: | | |  | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
| Project |  | | Date |  | Number |  | Type |  | Inject |  | Remove |  | Fix Time |  | Fix Ref. |
| 1A |  | |  |  | 2 |  |  |  |  |  |  |  |  |  |  |
| Description: | | |  | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
| Project |  | | Date |  | Number |  | Type |  | Inject |  | Remove |  | Fix Time |  | Fix Ref. |
| 1A |  | |  |  | 3 |  |  |  |  |  |  |  |  |  |  |
| Description: | | |  | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
| Project |  | | Date |  | Number |  | Type |  | Inject |  | Remove |  | Fix Time |  | Fix Ref. |
| 1A |  | |  |  | 4 |  |  |  |  |  |  |  |  |  |  |
| Description: | | |  | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
| Project |  | | Date |  | Number |  | Type |  | Inject |  | Remove |  | Fix Time |  | Fix Ref. |
| 1A |  | |  |  | 5 |  |  |  |  |  |  |  |  |  |  |
| Description: | | |  | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
| Project |  | | Date |  | Number |  | Type |  | Inject |  | Remove |  | Fix Time |  | Fix Ref. |
| 1A |  | |  |  | 6 |  |  |  |  |  |  |  |  |  |  |
| Description: | | |  | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
| Project |  | | Date |  | Number |  | Type |  | Inject |  | Remove |  | Fix Time |  | Fix Ref. |
| 1A |  | |  |  | 7 |  |  |  |  |  |  |  |  |  |  |
| Description: | | |  | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
| Project |  | | Date |  | Number |  | Type |  | Inject |  | Remove |  | Fix Time |  | Fix Ref. |
| 1A |  | |  |  | 8 |  |  |  |  |  |  |  |  |  |  |
| Description: | | |  | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |

# PSP Defect Recording Log

|  |  |
| --- | --- |
| Defect Types |  |
| 10 Documentation | 60 Checking |
| 20 Syntax | 70 Data |
| 30 Build, Package | 80 Function |
| 40 Assignment | 90 System |
| 50 Interface | 100 Environment |

|  |  |  |  |
| --- | --- | --- | --- |
| Student |  | Date |  |
| Program |  | Program # | 1A |
| Instructor |  | Language | C++ |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Project |  | | Date |  | Number |  | Type |  | Inject |  | Remove |  | Fix Time |  | Fix Ref. |
| 1A |  | |  |  | 9 |  |  |  |  |  |  |  |  |  |  |
| Description: | | |  | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
| Project |  | | Date |  | Number |  | Type |  | Inject |  | Remove |  | Fix Time |  | Fix Ref. |
| 1A |  | |  |  | 10 |  |  |  |  |  |  |  |  |  |  |
| Description: | | |  | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
| Project |  | | Date |  | Number |  | Type |  | Inject |  | Remove |  | Fix Time |  | Fix Ref. |
| 1A |  | |  |  | 11 |  |  |  |  |  |  |  |  |  |  |
| Description: | | |  | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
| Project |  | | Date |  | Number |  | Type |  | Inject |  | Remove |  | Fix Time |  | Fix Ref. |
| 1A |  | |  |  | 12 |  |  |  |  |  |  |  |  |  |  |
| Description: | | |  | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
| Project |  | | Date |  | Number |  | Type |  | Inject |  | Remove |  | Fix Time |  | Fix Ref. |
| 1A |  | |  |  | 13 |  |  |  |  |  |  |  |  |  |  |
| Description: | | |  | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
| Project |  | | Date |  | Number |  | Type |  | Inject |  | Remove |  | Fix Time |  | Fix Ref. |
| 1A |  | |  |  | 14 |  |  |  |  |  |  |  |  |  |  |
| Description: | | |  | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
| Project |  | | Date |  | Number |  | Type |  | Inject |  | Remove |  | Fix Time |  | Fix Ref. |
| 1A |  | |  |  | 15 |  |  |  |  |  |  |  |  |  |  |
| Description: | | |  | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
| Project |  | | Date |  | Number |  | Type |  | Inject |  | Remove |  | Fix Time |  | Fix Ref. |
| 1A |  | |  |  | 16 |  |  |  |  |  |  |  |  |  |  |
| Description: | | |  | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |
|  | | | | | | | | | | | | | | | | |

# PSP Time Recording Log

|  |  |  |  |
| --- | --- | --- | --- |
| Student |  | Date |  |
| Program |  | Program # | 1A |
| Instructor |  | Language | C++ |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Project | Phase | Start Date and Time | Int. Time | Stop Date and Time | Delta Time | Comments |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

Document Markings

Copyright 2020 Carnegie Mellon University. All rights reserved.  
  
This material is based upon work funded and supported by the Department of Defense under Contract No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.  
  
Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Department of Defense.  
  
NO WARRANTY. THIS MATERIAL IS FURNISHED ON AN “AS-IS” BASIS WITH NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, ANY WARRANTY WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT, OR THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.  
  
[Distribution Statement A] This material has been approved for public release and unlimited distribution. The United States Government has Unlimited Rights in this material as defined by DFARS 252.227-7013.

The text and illustrations in this material are licensed by Carnegie Mellon University under a [Creative Commons Attribution 4.0 International License](https://creativecommons.org/licenses/by/4.0/).

The Creative Commons license does not extend to logos, trade marks, or service marks of Carnegie Mellon University.