PSP Time Recording Log

Student _____ Date _____
Program _____ Program # _____
Instructor _____ Language _____

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

Time Recording Log Instructions

Purpose	- Use this form to record the time you spend on each project activity. - For the PSP, phases often have only one activity; larger projects usually have multiple activities in a single process phase. - These data are used to complete the Project Plan Summary. - Keep separate logs for each program.
General	- Record all of the time you spend on the project. - Record the time in minutes. - Be as accurate as possible. - If you need additional space, use another copy of the form. - If you forget to record the starting, stopping, or interruption time for an activity, promptly enter your best estimate.
Header	- Enter your name and the date. - Enter the program name and number. - Enter the instructor's name and the programming language you are using.
Project	Enter the program name or number.
Phase	Enter the name of the phase for the activity you worked on, e.g. Planning, Design, Test.
Start Date and Time	Enter the date and time when you start working on a process activity.
Interruption Time	- Record any interruption time that was not spent on the process activity. - If you have several interruptions, enter their total time. - You may enter the reason for the interrupt in comments.
Stop Date and Time	Enter the date and time when you stop working on that process activity.
Delta Time	Enter the clock time you actually spent working on the process activity, less the interruption time.
Comments	Enter any other pertinent comments that might later remind you of any unusual circumstances regarding this activity.

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 # ____________
Instructor ____________________ Language ____________

Project	Date	Number	Type	Inject	Remove	Fix Time	Fix Ref.

Description: ____________________

Project	Date	Number	Type	Inject	Remove	Fix Time	Fix Ref.

Description: ____________________

Project	Date	Number	Type	Inject	Remove	Fix Time	Fix Ref.

Description: ____________________

Project	Date	Number	Type	Inject	Remove	Fix Time	Fix Ref.

Description: ____________________

Project	Date	Number	Type	Inject	Remove	Fix Time	Fix Ref.

Description: ____________________

Project	Date	Number	Type	Inject	Remove	Fix Time	Fix Ref.

Description: ____________________

Project	Date	Number	Type	Inject	Remove	Fix Time	Fix Ref.

Description: ____________________

Project	Date	Number	Type	Inject	Remove	Fix Time	Fix Ref.

Description: ____________________

PSP Defect Recording Log Instructions

Purpose	- Use this form to hold data on the defects that you find and correct. - These data are used to complete the Project Plan Summary form.
General	- Record each defect separately and completely. - If you need additional space, use another copy of the form.
Header	- Enter your name and the date. - Enter the program name and number. - Enter the instructor's name and the programming language you are using.
Project	- Give each program a different name or number. - For example, record test program defects against the test program.
Date	Enter the date on which you found the defect.
Number	- Enter the defect number. - For each program or module, use a sequential number starting with 1 (or 001, etc.).
Type	- Enter the defect type from the defect type list summarized in the top left corner of the form. - Use your best judgment in selecting which type applies.
Inject	- Enter the phase when this defect was injected. - Use your best judgment.
Remove	Enter the phase during which you fixed the defect. (This will generally be the phase when you found the defect.)
Fix Time	- Enter the time that you took to find and fix the defect. - This time can be determined by stopwatch or by judgment.
Fix Ref.	- If you or someone else injected this defect while fixing another defect, record the number of the improperly fixed defect. - If you cannot identify the defect number, enter an X.
Description	Write a succinct description of the defect that is clear enough to later remind you about the error and help you to remember why you made it.

PSP Defect Type Standard

Type Number	Type Name	Description
10	Documentation	Comments, messages
20	Syntax	Spelling, punctuation, typos, instruction formats
30	Build, Package	Change management, library, version control
40	Assignment	Declaration, duplicate names, scope, limits
50	Interface	Procedure calls and references, I/O, user formats
60	Checking	Error messages, inadequate checks
70	Data	Structure, content
80	Function	Logic, pointers, loops, recursion, computation, function defects
90	System	Configuration, timing, memory
100	Environment	Design, compile, test, or other support system problems

PSP0.1 Process Script

Purpose	To guide the development of module-level programs
Entry Criteria	- Problem description - PSP0.1 Project Plan Summary form - Time and Defect Recording logs - Defect Type, ***Coding, and Size Measurement*** standards - Stopwatch (optional)

Step	**Activities**	**Description**
1	Planning	- Produce or obtain a requirements statement. - ***Estimate the added and modified size of this program.*** - Estimate the required development time. - Enter the plan data in the Project Plan Summary form. - Complete the Time Recording log.
2	Development	- Design the program. - Implement the design. - Compile the program, and fix and log all defects found. - Test the program, and fix and log all defects found. - Complete the Time Recording log.
3	Postmortem	Complete the Project Plan Summary form with actual time, defect, and size data.

Exit Criteria	- A thoroughly tested program - Completed Project Plan Summary form with estimated and actual data - ***Completed PIP forms*** - Completed Time and Defect Recording logs

PSP0.1 Planning Script

Purpose	To guide the PSP planning process
Entry Criteria	- Problem description - PSP0.1 Project Plan Summary form - Time Recording log

Step	**Activities**	**Description**
1	Program Requirements	- Produce or obtain a requirements statement for the program. - Ensure that the requirements statement is clear and unambiguous. - Resolve any questions.
2	***Size Estimate***	***Make your best estimate of the added and modified size of this program.***
3	Resource Estimate	- Make your best estimate of the time required to develop this program. - ***Using the To Date % from the most recently developed program as a guide, distribute the development time over the planned project phases.***

Exit Criteria	- Documented requirements statement - Completed Project Plan Summary form with estimated ***program size and*** development time data - Completed Time Recording log

PSP0.1 Development Script

Purpose	To guide the development of small programs
Entry Criteria	- Requirements statement - Project Plan Summary form with estimated program ***size and*** development time - Time and Defect Recording logs - Defect Type standard ***and Coding standard***

Step	Activities	Description
1	Design	- Review the requirements and produce a design to meet them. - Record in the Defect Recording log any requirements defects found. - Record time in the Time Recording log.
2	Code	- Implement the design ***following the Coding standard.*** - Record in the Defect Recording log any requirements or design defects found. - Record time in the Time Recording log.
3	Compile	- Compile the program until there are no compile errors. - Fix all defects found. - Record defects in the Defect Recording log. - Record time in the Time Recording log.
4	Test	- Test until all tests run without error. - Fix all defects found. - Record defects in the Defect Recording log. - Record time in the Time Recording log.

Exit Criteria	- A thoroughly tested program ***that conforms to the Coding standard*** - Completed Time and Defect Recording logs

PSP0.1 Postmortem Script

Purpose	To guide the PSP postmortem process
Entry Criteria	- Problem description and requirements statement - Project Plan Summary form with program size and development time data - Completed Time and Defect Recording logs - A tested and running program ***that conforms to the coding and size measurement standards***

Step	Activities	Description
1	Defect Recording	- Review the Project Plan Summary to verify that all of the defects found in each phase were recorded. - Using your best recollection, record any omitted defects.
2	Defect Data Consistency	- Check that the data on every defect in the Defect Recording log are accurate and complete. - Verify that the numbers of defects injected and removed per phase are reasonable and correct. - Using your best recollection, correct any missing or incorrect defect data.
3	***Size***	- ***Count the size of the completed program.*** - ***Determine the size of the base, reused, deleted, modified, added, total, added and modified, and new reusable code.*** - ***Enter these data in the Project Plan Summary form.***
4	Time	- Review the completed Time Recording log for errors or omissions. - Using your best recollection, correct any missing or incomplete time data.

Exit Criteria	- A thoroughly tested program ***that conforms to the coding and size measurement standards*** - Completed Project Plan Summary form - ***Completed PIP forms describing process problems, improvement suggestions, and lessons learned*** - Completed Time and Defect Recording logs

PSP0.1 Project Plan Summary

Student		Date	
Program		Program #	
Instructor		Language	

Program Size	*Plan*	*Actual*	*To Date*
Base (B)		*(Measured)*	
Deleted (D)		*(Counted)*	
Modified (M)		*(Counted)*	
Added (A)		*(T − B + D − R)*	
Reused (R)		*(Counted)*	
Added and Modified (A+M)		*(A + M)*	
Total Size (T)		*(Measured)*	
Total New Reusable			

Time in Phase (min.)	*Plan*	**Actual**	**To Date**	**To Date %**
Planning				
Design				
Code				
Compile				
Test				
Postmortem				
Total				

Defects Injected	**Actual**	**To Date**	**To Date %**
Planning			
Design			
Code			
Compile			
Test			
Total Development			

Defects Removed	**Actual**	**To Date**	**To Date %**
Planning			
Design			
Code			
Compile			
Test			
Total Development			
After Development			

PSP0.1 Plan Summary Instructions

Purpose	To hold the plan and actual data for programs or program parts
General	- ***Use the most appropriate size measure, either LOC or element count.*** - "To Date" is the total actual to-date values for all products developed.
Header	- Enter your name and the date. - Enter the program name and number. - Enter the instructor's name and the programming language you are using.
Program Size	- ***Enter the plan added and modified size value (A+M).*** - ***Enter actual base, deleted, modified, reused, total, and new reusable size.*** - ***Calculate actual added size as T-B+D-R and actual added and modified size as A+M.*** - ***Enter to-date reused, added and modified, total, and new reusable size.***
Time in Phase	- Enter the estimated total development time. - ***Distribute the estimated total time across the development phases according to the To Date % for the most recently developed program.*** - Enter the actual time by phase and the total time. - To Date: Enter the sum of the actual times for this program plus the to-date times from the most recently developed program. - To Date %: Enter the percentage of to-date time in each phase.
Defects Injected	- Enter the actual defects by phase and the total actual defects. - To Date: Enter the sum of the actual defects injected by phase and the to-date values for the most recent previously developed program. - To Date %: Enter the percentage of the to-date defects injected by phase.
Defects Removed	- To Date: enter the actual defects removed by phase plus the to-date values for the most recent previously developed program. - To Date %: Enter the percentage of the to-date defects removed by phase. - After development, record any defects subsequently found during program testing, use, reuse, or modification.

PSP Process Improvement Proposal (PIP)

Student		Date	
Program		Program #	
Instructor		Language	

Problem Description

Briefly describe the problems that you encountered.

Proposal Description

Briefly describe the process improvements that you propose.

Other Notes and Comments

Note any other comments or observations that describe your experiences or improvement ideas.

PSP Process Improvement Proposal (PIP) Instructions

Purpose	- To provide a way to record process problems and improvement ideas - To provide an orderly record of your process improvement ideas - To record any other noteworthy observations
General	Use the PIP form to - record process improvement ideas as they occur to you - establish priorities for your improvement plans - describe lessons learned and unusual conditions Keep PIP forms on hand while using the PSP. - Record process problems even without proposed solutions. - Submit a PIP with each PSP assignment report.
Header	- Enter your name and the date. - Enter the program name and number. - Enter the instructor's name and the programming language you are using.
Problem Description	Briefly describe any problems or experiences that led to this PIP.
Proposal Description	Describe the proposed improvement as explicitly as possible.
Other Notes and Comments	Briefly describe any other observations or facts that would later help you to - remember what you did while writing this program - remember an idea for a future improvement - explain to your instructor something you did and why you did it

C++ Coding Standard

Purpose	To guide implementation of C++ programs
Program Headers	Begin all programs with a descriptive header.
Header Format	/**/ /* Program Assignment: the program number */ /* Name: your name */ /* Date: the date you started developing the program */ /* Description: a short description of the program and what it does */ /**/
Listing Contents	Provide a summary of the listing contents
Contents Example	/**/ /* Listing Contents: */ /* Reuse instructions */ /* Modification instructions */ /* Compilation instructions */ /* Includes */ /* Class declarations: */ /* CData */ /* ASet */ /* Source code in c:/classes/CData.cpp: */ /* CData */ /* CData() */ /* Empty() */ /**/
Reuse Instructions	- Describe how the program is used: declaration format, parameter values, types, and formats. - Provide warnings of illegal values, overflow conditions, or other conditions that could potentially result in improper operation.
Reuse Instruction Example	/**/ /* Reuse instructions */ /* int PrintLine(char *line_of_character) */ /* Purpose: to print string, 'line_of_character', on one print line */ /* Limitations: the line length must not exceed LINE_LENGTH */ /* Return 0 if printer not ready to print, else 1 */ /**/
Identifiers	Use descriptive names for all variable, function names, constants, and other identifiers. Avoid abbreviations or single-letter variables.
Identifier Example	Int number_of_students; /* This is GOOD */ Float: x4, j, ftave; /* This is BAD */
Comments	- Document the code so 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	If(record_count > limit) /* have all records been processed? */
Bad Comment	If(record_count > limit) /* check if record count exceeds limit */
Major Sections	Precede major program sections by a block comment that describes the processing done in the next section.
Example	/**/ /* The program section examines the contents of the array 'grades' and calcu- */ /* lates the average class grade. */ /**/
Blank Spaces	- Write programs with sufficient spacing so they do not appear crowded. - Separate every program construct with at least one space.

(continued)

C++ Coding Standard (continued)

Indenting	- Indent each brace level from the preceding level. - Open and close braces should be on lines by themselves and aligned.

Indenting Example	while (miss_distance > threshold)
	{
	success_code = move_robot (target _location);
	if (success_code == MOVE_FAILED)
	{
	printf(“The robot move has failed.\n”);
	}
	}

Capitalization	- Capitalize all defines. - Lowercase all other identifiers and reserved words. - To make them readable, user messages may use mixed case.
Capitalization Examples	#define DEFAULT-NUMBER-OF-STUDENTS 15 int class-size = DEFAULT-NUMBER-OF-STUDENTS;

