**PSP2 Design Review Checklist**

|  |  |  |  |
| --- | --- | --- | --- |
| Student |  | Date |  |
| Program |  | Program # |  |
| Instructor |  | Language |  |

|  |  |
| --- | --- |
| Purpose | To guide you in conducting an effective design 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 design covers all of the applicable requirements.   * All specified outputs are produced. * All needed inputs are furnished. * All required includes are stated. |  |  |  |  |
| External Limits | Where the design assumes or relies upon external limits, determine if behavior is correct at nominal values, at limits, and beyond limits. |  |  |  |  |
| Logic | * Verify that program sequencing is proper.   Stacks, lists, and so on are in the proper order.  Recursion unwinds properly.   * Verify that all loops are properly initiated, incremented, and terminated. * Examine each conditional statement and verify all cases. |  |  |  |  |
| Internal Limits | Where the design assumes or relies upon internal limits, determine if behavior is correct at nominal values, at limits, and beyond limits. |  |  |  |  |
| Special Cases | * Check all special cases. * Ensure proper operation with empty, full, minimum, maximum, negative, and ero values for all variables. * Protect against out-of-limits, overflow, and underflow conditions. * Ensure “impossible” conditions are absolutely impossible. * Handle all possible incorrect or error conditions. |  |  |  |  |
| Functional Use | * Verify that all functions, procedures, or methods are fully understood and properly used. * Verify that all externally referenced abstractions are precisely defined. |  |  |  |  |
| System Considerations | * Verify that the program does not cause system limits to be exceeded. * Verify that all security-sensitive data are from trusted sources. * Verify that all safety conditions conform to the safety specifications. |  |  |  |  |
| Names | Verify that   * all special names are clear, defined, and authenticated * the scopes of all variables and parameters are self-evident or defined * all named items are used within their declared scopes |  |  |  |  |
| Standards | Ensure that the design conforms to all applicable design standards. |  |  |  |  |

**Code Review Checklist**

|  |  |  |  |
| --- | --- | --- | --- |
| Student |  | Date |  |
| Program |  | Program # |  |
| 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 |  |  |  |  |