[Date]

[Document title]

[Document subtitle]

Alexis Barltrop

Maggie Sun

Percy Hu

Tyler Baker

# Executive Summary

PandaTip: The executive summary should be between 150-600 words and should summarise the remainder of the report without preempting it. It is usually written last of all when you have a clear idea of the proposal as a whole. The aim of this section is to engage the readers’ interest and encourage them to read the rest of the software development proposal. You should include a helicopter view of the project including the software project name and objectives as well as a general mention of the technology. Don’t go into too much detail or include price (as both are included elsewhere). Also remember that the reader may not be technically savvy so avoid losing their interest with too much technical detail which can be included later.

# Contents

Contents

[Executive Summary 1](#_Toc477178156)

[Contents 0](#_Toc477178157)

[Introduction 1](#_Toc477178158)

[Project Description 2](#_Toc477178159)

[Resource Requirements 6](#_Toc477178160)

[Organisation 9](#_Toc477178161)

[Project Break Down 10](#_Toc477178162)

[Risk Analysis 14](#_Toc477178163)

[Project Schedule 15](#_Toc477178164)

[Monitoring and Reporting 17](#_Toc477178165)

[Conclusion 18](#_Toc477178166)

# Introduction

* No more than a page

Extend sitting time or sedantry behaviour is becoming one of the largest risk factors for office workers and people in developed countries. More and more of our lives is spent using technology specifically personal computers, watching television and other activities involving smart phones or tables which are generally done in a seated or sedantry position. The effect of prolonged periods of sedantry behaviour has been shown to be detrimental to health in a number of different ways. Studies citing it as a risk factor in heart disease, type 2 diabetes and cancer are increasingly showingthat the more time spent sitting down the more likely one of this conditions is going to develop or the worse it will be.

The solution to this problem is again a problem of habit forming? The solution is to simply take breaks at regular intervals. Correct posture and develop an ergonomic work environment. A simple a task as walking around the office for 1-2 minutes allows the body to get out of a sedantry behaviour and reduce the chance of back injury and other related injuries.

Regular breaks can also improve productivity and assist in concentration. The pomodoro technique favors productive working phases of 20-25 minutes followed by small breaks of 3-4 minutes. Every fourth or fifth break is usually longer, around 20-30 minutes. Taking these breaks as an oppourtunity to stretch and move out of a sedentry position is good example of a habitiual behavour that can imporve both health and saftey in the work/study space and productivty as well.

In this report we propose and elaborate on the development of a smart phone application deisgned to remind and assist it users to take regular breaks from sitting while working at a computer workstation. This is an application that is desirable to all the group members who studying as computer science students spend a majority of our time at a computer. It should be possible for use all to test it before we deliver the final program to the public.

The external constraints on this project are given in the assignment brief. The project must be written mostly in C++ or C, must be completed before a absolute deadline, the software must be something that we would use everyday and we must maintain it for at least a semester.

Cite some studies to look real professional

$$$ What citation method?

Introduce term for ergonomic breaks

Pomodoro

<https://en.wikipedia.org/wiki/Pomodoro_Technique>

https://www.focusboosterapp.com

Sitting down

<http://www.health.harvard.edu/blog/much-sitting-linked-heart-disease-diabetes-premature-death-201501227618>

Ergonomic calculator

http://www.ergotron.com/tools/workspace-planner

# Project Description

### Purpose

This aim of this project is to design and build a software product that aids in the formation of habits that will improve the quality of life of the user. For example, taking regular breaks during extended periods of computer usage to reduce back pain.

Definitions?

### User Description

The software product is primarily aimed at the office desk worker. A user of this type typically spends most their productive time, at work or in study, sitting down reading on paper or monitor screen. This type of user is most likely to benefit from reminders to take breaks for the health so they can focus on the task at hand. The work done is varied and there may be times when taking a break from an activity is not possible, or a distraction which removes them from their normal workflow. The user also has access to a smartphone, which we will be user to run the software product.

### Platform

The software product will be designed to run on a smart phone device, specifically the Apple iPhone range. The iPhone is consistently the highest selling smart phone device so there will be not shortage of people to use our App. The software will be written for iOS 10 which is supported by the iPhone models 7,6 etc. The limitations of these devices are discussed later. The app will be local and should not need to communicate with a server or cloud system to function completely. The user interface is then restricted to the touch screen input used on the iPhone.

### Description of functionality

#### Core Functionality

Recurring timer to alert user to take breaks

#### Extended Functionality

Create timer object for multiple different habits e.g. drinking water, taking breaks, doing stretches, taking supplements.

#### Functionality That Would be Super Cool to Have

Voice activated timer and timer setting.

### User Requirements

1. The software product will allow the user to set a timer for a custom length of time.
   1. The time for productive work.
   2. The time for break.
   3. Whether the 4th break is an extended one.

Rationale: Standard times for breaks and work do exist but the user is the best person to decide what their workflow will be like. We want to limit how long they work for, so that they don’t work to long without a break and we want to limit their break time so they don’t get distracted. The 4th or third break as an extended one comes from the Pomodoro method.

1. The software product will allow the user to start, pause and stop the timer

Rationale: If there is a distraction or likely change in the strict workflow set by the timer then the user should be able to pause all functions of the timer where they are and continue when get back to their desk. At the end of the day the user should be able to stop or turn off the timer completely.

1. The software product should alert the user when the work/break timers have finished
   1. The user should be able to select either a ring tone or vibrate tone.

Rationale: The software may be used in a quiet space and so the user should be able to use the vibrate function of phone to remind them. I may be possible for the start break and start work alerts to be different which may help with habit conditioning.

1. The app should be able to run in the background without impeding other activities on the phone.
   1. The app should either pause or mute during calls

Rationale: The app should require minimal user input and run in the background as much as possible. Alerts are undesirable during phone calls .

System Requirements

As per the requirements of this project all the objects and functions for this software must be written in C++ and only the user interface may be written in Swift.

Function

- Set Timer

* Start Timer
* Pause Timer
* Turn Timer On/Off

Description

Inputs

Source

Ouptus

Destination

Requires

Pre-condition

Post-condition

Side effects

### Specific Requirements

* External interface requirements
* Functional requirements
  + Program shall do…
* Performance requirements
  + Program shall be …

### Possibility for Expansion

With a working robust object type for the timer we should be able to create other object timers for other habitual actions. For example reminding to drink water during the day.

## Key Attributes

The software and any upgrades in functionality should exhibit the following four attributes:

### Usability

* The software should be intuitive and easy to use without a large amount of effort from the target user. The user interface should be appropriate for the platform and the target user. Adequate documentation will be kept available to instruct the user in situations where necessary.

### Dependability

* The nature of the project is one that we hope will improve the quality of the users life and health by managing the formation of good habits. To do this it must reliably remind the user to perform the habitual actions and only the habitual actions that are selected by the user. To miss a reminder or turn off reminders unexpectedly would count as a failure of the software to meet its purpose. To this end the user must be able to depend on the software to function without fault due to problems inherit with the software.

### Efficiency

* The software will be designed to operate in the background with only periodic interaction with the user. The mobile platform, while continuing to increase in capability, still has fixed limitations to its memory and processing power. The software should make efficient use of both memory and processor resources to ensure that other functions and software in the device is not unreasonably slowed down or made inoperable.

### Maintainability

* The premise of this project is to maintain the software for a period of 5 months in the second semester. In this time the functionality or layout of the software may change. Therefore the software should be written in a way such that it can change and ideally that it is easy to do so. In our group there is also the risk of having only one dedicated programmer proficient in C++. The software must be written and documented such that, in the case where the lead programmer is unavailable, another group member can understand and make constructive changes to the software.

# Resource Requirements

Hardware & software dependencies discussed here

## Hardware

The Apple iPhone maintains the highest market share of smart phones of any company today and has the app store for easy use of software applications. As a phone it is well suited to the purpose of this software as people always have their phones on them and thus will always be in range to be reminded to take a break. The iOS 10 operating system is compatible with iPhone models iPhone 7, iPhone 7 Plus, iPhone 6s, iPhone 6s Plus, iPhone 6, iPhone 6 Plus, iPhone SE, iPhone 5s, iPhone 5c, iPhone 5. ($$ Ref apple website <http://www.apple.com/nz/ios/ios-10/?afid=p238%7CsNQXC7DuD-dc_mtid_209259ru40377_pcrid_137662989248_&cid=wwa-nz-kwgo-features--slid->). Ashort description of their hardware is given below:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Storage Capacity / GB | Processor | RAM |
| iPhone 7 Plus | 32,128,256 | A10 Fusion chip with 64-bit architecture  Embedded M10 motion coprocessor  Max CPU clock rate 2.34 GHz | 3GB |
| iPhone 7 | 32,128,256 | A10 Fusion chip with 64-bit architecture  Embedded M10 motion coprocessor  Max CPU clock rate 2.34 GHz | 2GB |
| iPhone6s Plus | 32,128 | A9 chip with 64-bit architecture  Embedded M9 motion coprocessor  Max CPU clock rate 1.85 GHz | 2GB |
| iPhone 6s | 32,128 | A9 chip with 64-bit architecture  Embedded M9 motion coprocessor  Max CPU clock rate 1.85 GHz | 2GB |
| iPhone SE | 32,128 | A9 chip with 64-bit architecture  Embedded M9 motion coprocessor  Max CPU clock rate 1.85 GHz | 2GB |

Given the simple nature of the timer app we do not predict that there will be a large drain on memory or processor resources. While we aim to make an app which is efficient, the power of the hardware is such that small optimizations of the application will largely go unnoticed by the user.will take steps to reduce the size of the app but not a big deal.

Development requires computers running MacOS in order to use the iOs development kit and iPhone simulators in Xcode. In addition to the personal computers and Macbooks owned by the team members we also have access to computer suites at the university with many iMacs desktop computers available 24/7. Other than this we do not expect any other hardware to be needed for development.

The hardware requirements of a phone are small and not extreme. Therefore, it should not be a problem to do all development with an ordinary desktop computer.

## Software

All iPhone application will run on the iOS 10 operating system. As mentioned in the hardware section this OS is supported by the most recent iPhone models.

The application user interface will be designed and written in Swift code and the backend object and functions written in C++ as per the project requirements. The development and testing of the software will mostly use the Xcode application for MacOS. This application supports the development of apps sing Swift 3 and can also link to C++ objects.

The Xcode package manages the build of the application. This is done automatically when the application is run on the iPhone simulator in Xcode. If the build is done correctly and the application works as expected on the simulator then we assume that the application is valid for use on an actual iPhone. Xcode also uses a compiler that is optimized by Apple to work with iPhone CPUs helping achieve our target of efficiency.

Version control for the project will be implement using Git and github.com. While Xcode does have built in version control the requirements of this project mean that submission will be done using github and github is more universal. This would be a good opportunity for the members to learn how to use it. All the code in development and all deliverables will be made available on the Github repository of Percy Hu available at the following address:

<https://github.com/powerofpercy/COSC345-iOS-App/>

Communication through email

Using google drive for cloud storage and file sharing

Communication:

Discord (team chat)

Trello (for bugs and planned features and etc.)

Coding:

GUI:

Photoshop (paid, photo manipulation tool)

* Maggie has it on laptop. Available on computers in Compsci

Sketch (paid, education discount is available)

or other design tool (must have one, we will mainly design new icons and stuff instead of manipulating photos)

API docsets:

Dash (paid, very useful though)

# Organisation

Photos?

Percy Hu

Percy Hu is a third-year computer science student and the most experienced programmer in the group. Percy has experience programming in C++, C# etc. Percy takes the main responsibility for the implementation of the software and providing suggestions and direction in the planning phase.

Alexis Barltrop

Alexis is a Chemistry Masters Student turned Dip Grad Computer science student with minimal programming experience. The main role Alexis will be taking is that of writer and administrator. It will be Alexis responsibility to finalise reports and deliverables making them ready for submission as deliverables to the client. This includes the proposal and documentation. By the time the project is launched the goal is for Alexis to be proficient in C++ and able to assist Percy with maintenance.

Maggie

Maggie is a Psychology student turned dip grad student with minimal programming experience. Maggie has experience in graphic design and will be responsible for the appearance and design of all visual elements to the software including the user interface and user experience. This includes leading testing of the working application. Maggie will be gaining skills in C++ and Swift programming in order to work on the user interface and help the lead programmer in the maintenance phase.

* Designer and user experience controller?
* Lead Tester?

# Project Break Down

The project is broken down into 6 areas

1. Analysis

At the start of this project the idea was decided and the platform was chosen to be an iOS. The first part of analysis would be a feasibility study to determine whether or not the idea is possible. This will result in a feasibility report containing the follow areas:

* Examples of other applications doing similar things and their scope.
* A tutorial on linking swift and C++ to see how easy this is.
* Whether or not the project can be done with one trained programmer in a period of roughly four weeks.

At this stage we will investigate the software and hardware available to us to complete the project. This stage should be already well underway by the time this report is submitted.

Feasibility Study how do we attach swift and C++ objects to gether?

1. Development & Design

At this point the work is divided between Maggie design the user interface and Percy working on the code behind it.

Group meeting to decide on critical features required in both. Percy goes and writes the objects and functions required in C++. Maggie designs the interface in Swift. At the end we try and merge them together. While this is going on Alexis is monitoring documentation and making sure materials for final report being collected.

What are the constituent parts of the software?

1. Testing

Verification and Validation

Checklist code complete page 773

Testing on going using xCode smart phone simulator

* All group members should be able to do this using xCode

Then testing on smartphone platform

* All group members should be able to do this
* Check models of iPhones – want a number of different versions.

4. Deployment

Get onto App store after alpha testing? Get out as soon as possible.

Documentation milestones?

5. Maintenance

Quality Plan – how do we meet the key attributes?

Validation Plan

Maintenance Plan

Repair of Faults

Increase in Functionality

Adaption to new environments – Mac OSX?

Should have a software specification to define attributes and as a standard to check back on. – May be dynamic

Multi stage development with at least one update to the app store. By this stage at least one of the other group members should be able to

6. Training

Limited experience in software development only percy hu has done any games design before. So training and development is important in reducing risk.

Total estimation of man hours: 256

The deliverables provide will be as follows:

This report

Finished version ready for App store

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Task ID | Description of Task/Milestone | Estimated Days to Complete | People Assigned to Task | Dependencies |
| 1 | 345 |  |  |  |
| 1.1 | Analysis |  |  |  |
| 1.1.1 | Finalize Requirements for Software Product |  |  |  |
| 1.1.2 | Finalize Requirements for Development |  |  | 1.1.1 |
| 1.1.3 | Install Software & Hardware Needed for Development |  |  | 1.1.2 |
| 1.1.4 | Development Software & Hardware Installed & Working |  |  | 1.1.3 |
| 1.1.5 | Investigate Joining C++ to Swift |  | Percy/Tyler |  |
| 1.1.6 | Figure Out How to Get App Onto App Store |  | Alexis |  |
| 1.1.7 | Access to AppStore Developer Account |  |  | 1.1.6 |
| 1.1.8 | First Deliverable Project Proposal |  | Alexis |  |
| 1.2 | Development and Design |  |  |  |
| 1.2.1 | Design Class Library and Program Structure |  | Percy/Tyler |  |
| 1.2.2 | UML Diagram Complete |  |  | 1.2.1 |
| 1.2.3 | Design User Interface |  | Maggie |  |
| 1.2.4 | Agree on User Interface |  |  | 1.2.3 |
| 1.2.5 | Write Core Functionality in C++ |  | Percy/Tyler | 1.1.5, 1.2.2, 1.2.4 |
| 1.2.6 | Documentation Inspection #1 |  | Alexis |  |
| 1.2.7 | Write Extended Functionality #1 |  | Percy/Tyler | 1.2.5 |
| 1.2.8 | Documentation Inspection #2 |  |  |  |
| 1.2.9 | Create User Interface in Swift |  | Maggie , Percy/Tyler |  |
| 1.2.10 | Design App Thumb Nail |  | Maggie |  |
| 1.3 | Predeployment Testing |  |  |  |
| 1.3.1 | Test Core Functionality - Code Coverage |  | Alexis , Maggie | 1.2.5 |
| 1.3.2 | Test Core Functionality - User Interface |  | Alexis , Maggie | 1.2.5 |
| 1.3.3 | Test Extended Functionality #1 |  | Alexis , Maggie | 1.2.7 |
| 1.3.4 | Debugging and Suggested Changes From Testing |  | Percy/Tyler | 1.3.1, 1.3.2, 1.3.3 |
| 1.4 | Deployment |  |  |  |
| 1.4.1 | Launch App on Appstore |  |  | 1.3.4 |
| 1.4.2 | Second Deliverable: Completed Software |  |  | 1.4.1 |
| 1.5 | Training |  |  |  |
| 1.5.1 | Learning Swift - User Interface Design |  | Alexis , Maggie , Percy/Tyler |  |
| 1.5.2 | Learning C++ - Basic Coding |  | Alexis , Maggie |  |
| 1.5.3 | Tutorial on Swift/C++ Linking |  |  |  |
| 1.6 | Maintenance |  |  |  |
| 1.6.1 | Second Semester Begins |  |  |  |
| 1.6.2 | Assess Compatibility of Software to iOS/ iPhone 8¬† |  |  |  |
| 1.6.3 | Implement Changes for new iOS/ iPhone 8 |  |  |  |
| 1.6.4 | Identify Requirements for First Update |  |  |  |
| 1.6.5 | Write Extended Functionality #2 |  |  |  |
| 1.6.6 | Implement Patches/ Bug Fixes |  |  |  |
| 1.6.7 | Update App on App Store #1 |  |  |  |
| 1.7 | Ongoing Testing |  |  |  |
| 1.7.1 | Core/Extended #1 Functionality Testing - Extended Coverage |  |  |  |
| 1.7.2 | Extended Functionality # 2 Testing |  |  |  |
| 1.7.3 | Download¬† |  |  |  |

# Risk Analysis

A number of possible risks to the project are described in table $$XXX. The largest risk to this project lies in the inexperience of the group as a whole in developing software and writing in C++. Dealing with this is done with specialisation of roles. The two group members with experience in C/C++ are responsible for the implementation of the code. This should leave the other two members to take care of admin and design aspects of the project. Training and education for all group members is important to increase redundancy. People in the group may have other assessments due at critical times and may not be able to attend to important issues. Thus all members of the group aim to develop profienciy in C++ and Swift by the time the software is placed on the App Store. Maggie and Alexis have scheduled weekly time to develop these skills.

The risk of inexperience will be most present in the planning phase. It is easy to imagine that we may underestimate the time it takes to develop the software. Taking this into account we take an incrmental approach to design. The core funcitonality is the first thing to be implemented and extended funcitonality can be pushed back if the core development overshoots.

Version control and storing files on the cloud helps alleviate the risk of conflicting version of software from different people and also due to hardware failure. All files written on personal hardware must be backed up to the github repository or google drive. In the event that personal hardware fails, the back up is the copmuters in the computer science building. These are maintained by the department with staff and resources devoted to there operation. These machiens have mac and Xcode, however we can’t install other software on them.Thus most of communcation and management software a web platform and can be used without installing software.

Apple is expected to realease a new version of iOS and a new iPhone 8 later in September of 2017. Developer releases for this is expected in July 2017 and we aim to update our software to be compatible with these new releases. We don’t know what they will contain but expect a beta release of iOS 11 in June/July. We aim to address this compabiltiy in second semester before triyng ot extend the functionality further.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Risk ID | Risk | Description | Solution | Monitoring |
| 1. | The ease of joining Swift and C++ is more difficult than expected. | This is something no one in the group has done before |  |  |
| 2. | Percy ill or unavailable at a critical time during development |  |  |  |
| 3. | Changes to requirements changed dramatically |  |  |  |
| 4. | The time required to develop the software is underestimated |  |  |  |
| 5. | A new version of the iPhone is released for which we haven’t tested on. | Also released in september 2017 |  | Xcode update in july /september with iPhone 8 simulator. |
| 6. | A new version of iOS is released. | New verison of iOS expected in september 2017. Beta will be released in june. |  | Will need to sign up for the beta programme in July.  Put in dependecies. |
| 7. | The personal computer of group member breaks or is unavailable. |  |  | All documents on cloud to prevent loss. |
| 8. | The software/hardware at university is unavailable at a critical time |  |  |  |

Updates to iPhone and iOS may require extra testing.

Schedule for updates is known.

Largest risk in the development of this software is the small team size and lack of redundancy. Percy Hu is the only one with experience in C++. While the other members of the group have programming experience it will still take time to train up to a level of competence. The idea is to get everyone in the group acquainted with C++ ($$$Can define more concrete goals?) by second semester.

This also means that documentation must be complete so that we can understand if percy not there.

Risk on incoherence in project architecture?

Foreseeable problems and solutions

# Project Schedule

The project schedule is visualised in a Gannt chart in figure $$XX. From the submission date to the deadline for the completed software is 6 weeks. The functionalilty of the software has been cut into different levels and complexities. Thus the design phase task are flexible and the proposed schedule represents a best case scenario where everything is acheivable. Given the stric deadline it is morelikely that some functionaility will be lost but hopefully the incremental development will mean tis will not affect whether or not the software works.

A break down of the task is presented in table $$XX showing assigment, length of task and assignment.

WE aim to complete the core funcitonality and the add one extended function on top of that beofr ethe first deadline. In the maintenance part we aim to introduce another extend funcitonality ot he update and then one super cool functionality if possible.

The dead line for this project is the 29th May, approximately 6 weeks after the hand in of this report.

How big is the average iPhone app?

Going by COCOMO model. Given two months to complete the app the upper limit will be about 3000 lines of code using only one programmer.

Assignment of Tasks?

Put Gannt Chart in here

# Monitoring and Reporting

Monitoring

The most critical phase to monitor in the project is the design and development stage where the software will be written and tested. The approach to monitoring this stage is to break in down into a number of incremental stages which can be completed in at most a week. The TeamGannt web application will be used to report completion of each of these tasks. Durign the development stage Alexis is responsible for monitoring and inspection the code to make sure that style and documentation guidelines are consistent and up to standard for submission. A checklist for style and documentation is attached in the appendix.

For tasks in the analysis stage there are some things which are relevant to all group members and so the reporting for these will be done at a team meeting. Important knowledge such as the baiscs of linking swift and C++ will be taught to everybody in the group to reduce the risk of that knowledge being lost should one of the members become unavailible at a critical stage. Any other crictical information like this will be reported in a similar manner.

The small size of the group lends itself to face to face communication. The number of formal reports will be kept to any absolute minimum instead favouring open commmunication between the group. Discord chat client is being used to communicate and record chat. Formal meeting minutes may need to be kept.

In the testing and maintenance phases the important aspect will be in reporting of bugs and monitoring whether the bugs have been fixed. Ulitimately we want everybody to test the software. To this end we use Trello a web application. Bugs and related material i.e. screen shots, test data etc can be added here and made availible to the programmers. This software sends out notificaitons when any new bugs have been reported.

How the project will be monitored

When reports are to be delivered.

Reporting bugs – done using trello

Reporting whether tasks done using Team Gannt

Monitoring done via inspections milestones

Progress monitored by dividing up into managable chunks. Looking for milestones not percent completion.

Alexis to inspect documentation of code while being written

Use standard checklist to inspect code.

How are we going to monitor progress and what reporting will there be?

Milestones along the way.

Github repository available to all to check.

Reporting – Meetings, email etc

# Conclusion

Appendix – Checklists

COSC345 Commenting/Documentation Checklist

Style checklist

General

❑Is formatting done primarily to illuminate the logical structure of the code?

❑Can the formatting scheme be used consistently?

❑Does the formatting scheme result in code that’s easy to maintain?

❑Does the formatting scheme improve code readability?

Variables

* Do all variables have meaningful names?
* Are they used only for the purpose they are named?
* Are constants used instead of magic values?
* Are enumerated types used instead of encoded values?
* Any relationships between variables that could be simplified?
* Are variables declraed with correct scope?
* An global variables need to be local or vice versa?

Individual Statements

* Does each line contain at most one statement?
* Is each statement written without side effects?
* Is there at most one data declaration per line?
* Does error handling fix the problem or stop the program?
  + E.g. no continuation of code after catch blocks.
* Is white space used to make logical expressions, array references, and routine arguments readable?
* Do incomplete statements end the line in a way that’s obviously incorrect?
* Are continuation lines indented the standard indentation amount?

Functions/Routines/Methods

* Does each function perform one and only one well defined task?
* Does the name descrive exactly what the rountine does?
* Are the arguments to each routine formatted so that each argument is easy to read, modify, and comment?
* Are blank lines used to separate parts of a routine?

Control Structures

❑Does the code avoid doubly indented begin-end or {} pairs?

❑Are sequential blocks separated from each other with blank lines?

❑Are complicated expressions formatted for readability?

❑Are single-statement blocks formatted consistently?

❑Are case statements formatted in a way that’s consistent with the format- ting of other control structures?

❑Have gotos been formatted in a way that makes their use obvious?

Commenting Checklist

Comments

* Do the comments clarify the intent of the code rather than repeat or try to explain code?
* Do the comments summarise the code and help the reader?
* Are the comments indented the same number of spaces as the code they comment?
* Is the commenting style easy to maintain?
* Is there any disagreement between the code and the comments?
* Any redundent development markers?
* Any out of place single line comments?
* Any unexplained abbreaviations?
* Any parts that are uncommented and need to be?
* Have you been commenting as you write the code?

Variable comments

* Units and allowable ranges in comments?
* Comment limits on input data
* Comment why global variables global
* Comment any relationships between variables if they exist.

Routines

* Any third party routines documented with source?
* All non-trivial rountines commented?

Special Cases

* Document compiler work arounds
* Document any style violations

Classes, Files and Programs

❑Is there a one-to-one relationship between classes and files for most classes and files?

❑If a file does contain multiple classes, are all the routines in each class grouped together and is each class clearly identified?

❑Are routines within a file clearly separated with blank lines?

❑In lieu of a stronger organizing principle, are all routines in alphabetical sequence?