S1 Algorithms and Data Structures

The following passage is the introductory paragraph to a unit programming environments for motion, graphics and geometry. Read the passage and answer the questions that follow. Words likely to be unknown by your ELLs have been substituted with the word "huh"

A program can be designed with the **huh** of tools, paper and **huh**, or in the **huh** head. In the **huh** of such **huh** environments, a program design may contain **huh** concepts expressed in an **huh huh**. Before he or she can **huh** this program the **huh** needs a programming environment, typically a complex system with many distinct components: a computer and its operating system, **huh**, and program libraries; text and program editors; various programming languages and their **huh**. Such real programming environments force **huh** to express themselves in formal **huh**.

Programming is the **huh** of a solution to a problem, expressed in terms of those operations provided by a given programming environment. Most **huh** work in environments that provide very powerful operations and tools.

The more powerful a programming environment, the simpler the programming task, at least to the expert who has achieved **huh** of this environment. Even an experienced **huh** may need several months to master a new programming environment, and a **huh** may give up in **huh** at the **huh** of concepts and details he or she must understand before writing the simplest program the simpler a programming environment.

The easier it is to write and run small programs, and the more work it is to write substantial, useful programs. In the early days of computing, before the **huh** of programming languages during the 1960s, most **huh** worked in environments that were **huh** simple by modern standards: **huh** with an **huh** a **huh**, and a small program library **huh**. The programs they wrote were small compared to what a professional **huh** writes today. The simpler a programming environment is, the better suited it is for learning to program. **huh** today simple environments are hard to find! Even a home computer is **huh** with complex software that is not easily ignored or **huh**. For the sake of education it is useful to invent **huh** programming environments. Their only purpose is to illustrate some important concepts in the simplest possible setting and to **huh** insight. Part 1 of this book introduces such a toy programming environment suitable for programming **huh**, and motion and illustrates how it can gradually be **huh** to approach a simple but useful **huh** environment.

Questions:

1. **Are you confident that you summarize the main points of the paragraph?**

2. **Approximately how long did it take you to read this passage?**

3. **If you had to look up the huh words in a dictionary, approximately how long would it take you to read this passage?**

S1 Algorithms and Data Structures

Now read the passage with all the words visible.

A program can be designed with the barest of tools, paper and pencil, or in the programmer head. In the realm of such informal environments a program design may contain vague concepts expressed in an informal notation. Before he or she can execute this program the programmer needs a programming environment, typically a complex system with many distinct components: a computer and its operating system, utilities, and program libraries; text and program editors; various programming languages and their processors. Such real programming environments force programmers to express themselves in formal notations.

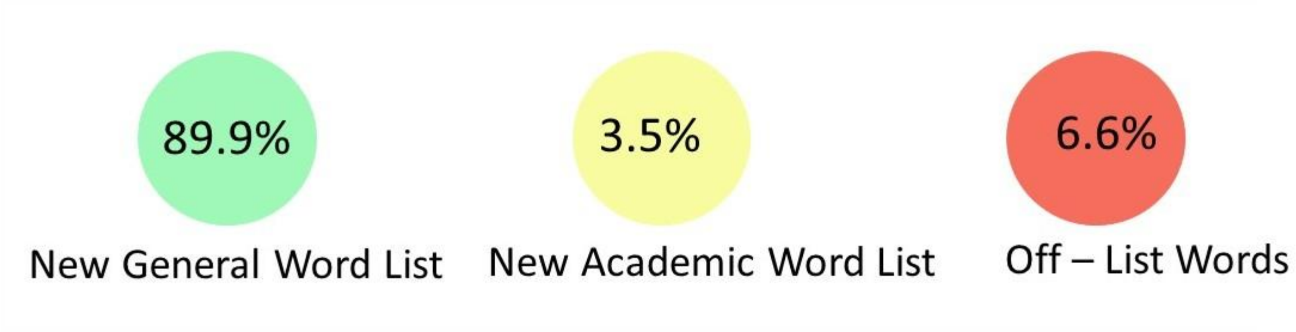
Programming is the realization of a solution to a problem expressed in terms of those operations provided by a given programming environment. Most programmers work in environments that provide very powerful operations and tools.

The more powerful a programming environment, the simpler the programming task, at least to the expert who has achieved mastery of this environment. Even an experienced programmer may need several months to master a new programming environment, and a novice may give up in frustration at the multitude of concepts and details he or she must understand before writing the simplest program the simpler a programming environment.

The easier it is to write and run small programs, and the more work it is to write substantial, useful programs. In the early days of computing, before the proliferation of programming languages during the 1960s, most programmers worked in environments that were exceedingly simple by modern standards: acquaintance with an assembler a loader, and a small program library sufficed. The programs they wrote were small compared to what a professional programmer writes today. The simpler a programming environment is the better suited it is for learning to program. Alas today simple environments are hard to find! Even a home computer is equipped with complex software that is not easily ignored or bypassed. For the sake of education it is useful to invent artificial programming environments. their only purpose is to illustrate some important concepts in the simplest possible setting and to facilitate insight. Part 1 of this book introduces such a toy programming environment suitable for programming graphics, and motion and illustrates how it can gradually be enriched to approach a simple but useful graphics environment.

Word Coverage

Percentages of words in each category for the passage you just read:



Much of the text belongs to the New General Service List which represents the 2800 core English words.

Some of the words are in the New Academic Word List which represents 963 words and their forms that are common in academic texts across disciplines.

artificial, execute, facilitate, informal, notation, novice, processors, realm, utilities, vague

---

Some of the words do not appear on either list. Many of these words are subject specific jargon that are important to understand the subject content.

acquaintance, alas, assembler, barest, bypassed, enriched, equipped, exceedingly, frustration, graphics, loader, mastery, multitude, pencil, programmer, proliferation, realization, sufficed

Most ELLs will know most of the common text found on the General Service List and some of the vocabulary from the New Academic Word List if they have studied those words or encountered them previously. They may be familiar with some of the off-list text also depending on their prior learning and their first language which may have some similar words (cognates) or similar root words. In order to read with good comprehension, readers should know 98% of the vocabulary in the text (Schmitt et al., 2011).

S1 Algorithms and Data Structures

Task A

The following passage is from the same unit on programming environments for motion, graphics and geometry. Highlight the words that you think would not be on the new general word list or the new academic word list in this passage from the section "Monopoly." Determine if the highlighted words should be "learned" or put in a glossary.

Procedures as Building Blocks

A program is built from components at many different levels of complexity. At the lowest level we have the constructs provided by the language we use: constants, variables, op erators, expressions, and simple unstructured statements. At the next higher level we h ave procedures: they let us refer to a program fragment of arbitrary size and complexity as a single entity, and build nested structures. Modern programming languages provide yet another level of packaging: modules or packages, useful for grouping related data a nd procedures. We limit our discussion to the use of procedures.
Programmers accumulate their own collection of useful program fragments. programmin g languages provide the concept of a procedure as the major tool for turning fragments i nto reusable building blocks. A procedure consists of two parts with distinct purposes: The heading specifies an important part of the procedure's external behavior through th e list of formal parameters: namely, what type of data moves in and out of the procedure . The body implements the action performed by the procedure, processing the input dat a and generating the output data.
A program fragment that embodies a single coherent concept is best written as a proce dure. This is particularly true if we expect to use this fragment again in a different contex t. The question of how general we want a procedure to be deserves careful thought. If th e procedure is too specific, it will rarely be useful. If it is too general it may be unwieldy: t oo large, too slow, or just too difficult to understand. The generality of a procedure depe nds primarily on the choice of formal parameters.

S1 Algorithms and Data Structures

Online tools such as the NAWL Highlighter can analyze text to identify words that are on the New Academic Word List, New General Service Word List or are Off-List. The online tool Rewordify identifies difficult vocabulary and provides simplifed wording. Teachers can use this tool to quickly identify words that may be difficult for ELLs. Based on the suggestions, text can easily be modified. The simplified wording can be used to quickly generate vocabulary tasks and glossaries. Students can use this tool to support their reading comprehension.

The following is text output from Rewordify. The text in brackets is the simplified text.

A program can be designed with the **barest of** [a very small amount of] tools, paper and pencil, or in the programmer head. In the **realm** [world] of such informal **environments** [surrounding conditions] a program design may contain **vague** [unclear] **concepts** [ideas] expressed in an informal **notation** [note/way of writing]. Before he or she can **execute** [run] this program the programmer needs a programming **environment** [surrounding conditions], **typically** [usually] a complex system with many **distinct** [clear/separate] **components** [parts/pieces]: a computer and its operating system, utilities, and program libraries; text and program editors; **various** [different] programming languages and their processors. Such real programming **environments** [surrounding conditions] force programmers to express themselves in formal **notations** [notes/ways of writing].

Programming is the **realization** [understanding/achieving a goal] of a solution to a problem expressed in terms of those operations provided by [given by] a given programming **environment** [surrounding conditions]. Most programmers work in **environments** [surrounding conditions] that provide very powerful operations and tools.

The more powerful a programming **environment** [surrounding conditions], the simpler the programming task, [job,] at least to the expert who has **achieved** [accomplished or gained with effort] mastery of this **environment** [surrounding conditions]. Even an experienced programmer may need **several** [more than two, but not a lot of] months to master a new programming **environment** [surrounding conditions], and a novice [beginner] may give up in frustration at the **multitude** [large number] of **concepts** [ideas] and details he or she must understand before writing the simplest program the simpler a programming **environment** [surrounding conditions].

S1 Algorithms and Data Structures

Tips

What can you do to support ELLs with assigned readings for your course?

1. Be selective in quantity and general accessibility
    - Select the minimum number of accessible readings that are essential for your course. Additional readings can be assigned as additional readings but all assignments and testing should be based on the essential readings.
    - Assign specific sections of a chapter to read rather than the whole chapter. Chose sections that align with your specific learning objectives. Give students a specific purpose for completing the reading so they know what they are supposed to learn from the reading.
2. Provide definitions of key subject specific vocabulary
    - Identify key terminology with simply written definitions from an ELL dictionary (https://www.oxfordlearnersdictionaries.com/us/) for your subject. Alternatively, create a class collaborative dictionary. Ask students which words are unfamiliar in the readings to develop an understanding of the challenges for your typical learners.
3. Highlight text features
    - Early in your course, take some time to highlight how to navigate your course textbook. Awareness of the usefulness of features such as the table of contents, glossary, index (important words often appear both in the glossary and the index) and appendices is beneficial for ELLs. Taking a "chapter walk" with your students illustrates helpful features of a text such as chapter objectives (to help identify key concepts), headings and subheadings, text (colour, italics and bold used to indicate important words), side notes, illustrations, graphics, captions, review questions, quizzes and further readings. Focus on how these features organize the text and highlight important information.
4. Scaffold Learning
    - Provide questions based on the text that focus on key language and concepts to help ELLs understand what you want them to learn from the reading. An advance reading organizer helps ELLS prioritize important vs unimportant details and also indicates when they have missed important pieces of information. For example, if the organizer asks them to list the four identifying features of a specific item and they can only list three, then they know that they need to re-read the passage or seek clarification from you.
5. Choose e-resources when possible
    - E-resources support the use of multiple tools that may help ELLs comprehend the text. Encourage your students to install Read Write Gold, a free toolbar. Its features include a phonetic spell checker, picture dictionary, text-to-speech, speech-to-text, translator, screen shot reader, vocabulary list builder, concept mapping, word prediction, PDF aloud, word banks, voice notation, and highlighting. With this tool, your ELLs can hear the text read aloud, find meanings of words easily and highlight key concepts.

S1 Algorithms and Data Structures

- Use the web app remodify.com to indicate vocabulary in a reading passage that might be challenging to comprehend, and to view accessible wording for ELLs. Encourage students to use the site to generate reading **passages with the difficult words replaced or defined** [insert link to highlighted passage] with more accessible language. Settings can be changed to view the original side by side with the same passage that has had the difficult words replaced by simplified language.

Put it in Action

1. Pick a selection from your course readings and highlight the subject specific vocabulary that may be unknown by ELLs in your courses.
2. If there is a text glossary – check to see if these words have been defined.
3. For any words not already defined, write a simple definition. Use a learner dictionary to find definitions that are accessible for ELLs (https://www.oxfordlearnersdictionaries.com/us/) or use rewordify.com to provide simplified wording.

Task A Answer

The following words are the words in the passage that are not on the New General Service Word List. Some of the words are frequently used in many contexts and appear on the New Academic Word List so students should learn them. Others are used specifically for this context. For those words, students should be given a reference glossary. Drag the words into the categories "learn" and "glossary."

accumulate, arbitrary, coherent, embodies, entity, generality, namely, nested,

parameters, programmers, reusable, unstructured, unwieldy

Learn: accumulate, arbitrary, coherent, entity,] namely, nested, parameters

Glossary: embodies, generality, programmers, reusable, unstructured, unwieldy