Date

CISC 205 – OOPS C++

Professor Larry Forman

Telephone: 619.388.3666

E-Mail: [LForman@sdccd.edu](mailto:lforman@sdccd.net)

Mail Box: Room A-8

Office: BT-210-G

**TRAINING ASSIGNMENT #1.4AI: AFFIRMATIVE INTERACTION**

**=== DUE ===**

**=============**

**TASKS:**

**0 –** First, read this Task Sheet!

Second, put a check mark by each Task number **and** letter when you complete it.

**1 –** **READ**: **HO#1.4, P#1TT, TDB**

**2 –** **TA OBJECTIVES**

* Develop a menu-based C++ program to produce interactive screen output
* Incorporate vectors and file I/O
* Maintain effective Michelangelo documentation

**3 – BACKGROUND / SPECIFICATIONS**

“Artificial intelligence” was coined in 1955 by John McCarthy at MIT (before he joined Stanford) to mean “the science and engineering of making intelligent machines”. Since then, the field has burgeoned in unexpected ways. In the early years, astonishing successes emerged in AI. However, it eventually became apparent that what people thought might be hard to do in AI, actually became easy to do with AI. On the other hand, what people thought might be easy to do in AI actually became very hard to do in AI.

One of the early successes was Joseph Weizenbaum’s computer program called ELIZA. This MIT professor’s early effort at natural language processing was named after Eliza Doolittle, a character in George Bernard Shaw’s play Pygmalion (which gave rise to the musical, My Fair Lady by Alan Jay Lerner and Frederick Loewe). The working class, Cockney flower girl Eliza Doolittle was taught to speak “proper” English so that she might pass as a refined lady I merry old England. Weizenbaum’s computer program grew out of the earlier program, DOCTOR. This was meant to simulate the conversation between a patient and a psychotherapist who uses the person-centered therapy developed by Carl Rogers in the 1940’s. Instead of having a database of real-world knowledge, DOCTOR would respond very generally to remarks of a patient. For example, a comment of “my leg hurts” might elicit DOCTOR’s reply of “what else hurts?” or “why do you say your leg hurts?” ELIZA employed a more general pattern matching approach. Surprisingly, some people treated ELIZA’s responses quite seriously and imbued the computer program with very human-like insights.

**STAR(1+): Research the history of artificial intelligence and present a PowerPoint presentation highlighting the origins and intriguing facts and tidbits about AI and pioneers in the field.**

**STAR(1+): Extend previous STAR by adding “pizzazz” to the PowerPoint presentation with dynamic graphics, transitions, audio and possibly relevant video**

In the spirit of ELIZA, create a program, LarrysAI (no spaces or punctuation), that is taught synonyms for “yes”. A vector will store the synonyms of the current session while an evolving “dictionary” will be written to and read from disk. Your project will include each of the following as separate functions with self-descriptive names and appropriate modifications – in your main(), where you deploy ONLY custom-defined functions along with any necessary local variables and constants. **BTW: NO GLOBAL VARIABLES ALLOWED**:

1. The "Welcome" function with your name via a **const** to greet the user.
2. The “Menu” function with appropriate error-handling of “dirty data” from user. Assume replies are NOT case-sensitive so lower- and uppercase values are valid
3. A separate function for EACH menu item: Sign-in, Logo, Affirmative Interaction, Exit (etc. for some STARS)
4. “Nest” a "Hit ENTER to Continue" message function in any of the above items to enhance readability. But do NOT deploy your hitEnter function in the main.

**4** – **SCREEN OUTPUT**

Display the following with appropriate blank lines between each section:

Welcome to Larry’s Affirmative Interaction Program

<<< Hit ENTER to continue >>>

MENU

S – Sign in

L – Display Logo

A – Start Affirmative Interaction

D – Display Dictionary

X – Exit Program

Please enter your selection and hit RETURN: z

Sorry, but “z” is not a valid entry.

Time to try again . . .

**(NOTE: Do complete “error-handling” of “dirty” data)**

MENU

S – Sign in

L – Display Logo

A – Start Affirmative Interaction

D – Display Dictionary

X – Exit Program

Please enter your selection and hit RETURN: s

Thank you for selecting “s” . . .

Time to sign in!

Please sign in with your first name: Albert

Ah, Albert, so nice you have you join us for

a little affirmative interaction . . .

MENU

S – Sign in

L – Display Logo

A – Start Affirmative Interaction

D – Display Dictionary

X – Exit Program

Please enter your selection and hit RETURN: L Thank you for selecting “L” . . .

Time for Larry’s logo!

(Display your logo from TA #1.2Q)

<<< Hit ENTER to continue >>>

MENU

S – Sign in

L – Display Logo

A – Start Affirmative Interaction

D – Display Dictionary

X – Exit Program

Please enter your selection and hit RETURN: a

Thank you for selecting “a” . . .

Time for some affirmative interactions!

Let me tell you what’s already in my dictionary:

yes, YES (and I also know no and NO for . . . no)

Can you please teach me another word that means “yes”? yes

Good – please enter a word that means “yes”: Ok

Thanks – I will add “Ok” to my dictionary.

Can you please teach me another word that means “yes”? Ok

Good – please enter a word that means “yes”: suRe

Thanks – I will add “suRe” to my dictionary

Can you please teach me another word that means “yes”? Ok

Good – please enter a word that means “yes”: Ok

Thanks – but I already know “Ok”.

Can you please teach me another word that means “yes”? yup

Hmm, I don’t know “yup”. Does it mean “yes”? Ok

Thanks – I will add “yup” to my dictionary

Can you please teach me another word that means “yes”? NO

Well, thanks for your positive, never-say-no attitude! Glad you believe in affirmative interaction! In this session I learned the following new synonyms for yes:

Ok suRe yup

Now, I will update my disk file, LarrysYES\_Dictionary, with the new synonyms.

**NOTE: Plan to deal with the following scenario:**

Can you please teach me another word that means “yes”? yup

Hmm, I don’t know “yup”. Does it mean “yes”? uhuh

Hmm, I don’t know “uhuh”. Does it mean “yes”? si

Hmm, I don’t know “si”. Does it mean “yes”? certainly

Hmm, I don’t know “certainly”. Does it mean “yes”? ok

Thanks – I will add “yup” to my dictionary

**STAR: In the above scenario ALSO add the replies to “Hmm, …” – like uhuh, si, certainly – to the dictionary of synonyms**

**STAR: In the above STAR, do not add duplicates. For example:**

**Can you please teach me another word that means “yes”? yup**

**Hmm, I don’t know “yup”. Does it mean “yes”? uhuh**

**Hmm, I don’t know “uhuh”. Does it mean “yes”? si**

**Hmm, I don’t know “si”. Does it mean “yes”? uhuh**

**Hmm, I don’t know “certainly”. Does it mean “yes”? ok**

**Thanks – I will add “yup, uhuh, si” to my dictionary**

**STAR: Expand the previous STAR to respond with “Hey, I already told you I don’t know what “uhuh” means, etc.**

**STAR: Allow for case-insensitivity for all entries, so “suRe”, “sure”, “SURE”, etc. are all equivalent**

**STAR: Read a line, not just one word replies etc. in order to handle phrases like “you betcha”, “of course”, etc.**

**STAR: Add a menu items to add synonyms of “NO” to a dictionary and to display the “NO” dictionary items from disk**

**STAR: Extend previous STAR to recognize all of the “NO” synonyms when adding “YES” synonyms to the “YES” dictionary.**

**STAR: When adding “NO” synonyms to a dictionary, recognize all “YES” synonyms that are in the “YES” dictionary**

**STARS(1 or more): Add menu items for any of the following features:**

* **Search for a specific synonym in the dictionary and reply “found” or “not found”, as is appropriate**
* **Remove a specific synonym from the dictionary and report the update has been successfully completed (or not)**
* **Sort the dictionary alphabetically**
* **Reverse the order of the items in the dictionary**
* **Display a “word count” of how frequently each synonym appears during all the “A” sessions**
* **Display the most frequently used synonym for “yes” used during all the “A” sessions and indicate the number of times it was used**
* **Expand the previous feature to list the frequencies (and names) of all the synonyms used is sessions from highest to lowest frequency**
* **Search for duplicate entries in the dictionary and alert user about the outcome**

**STAR: Create (and demonstrate) scenarios that “crash” the system. Outline strategies to successfully handle these scenarios.**

**STARS(1 or more): Successfully implement any of the strategies in the previous STAR**

<<< Hit ENTER to continue >>>

MENU

S – Sign in

L – Display Logo

A – Start Affirmative Interaction

D – Display Dictionary

X – Exit Program

Please enter your selection and hit RETURN: d

Thank you for selecting “d” . . .

Time to display my dictionary with only synonyms for “yes”

Let me tell you what’s already in my dictionary:

yes, YES (and I also know no and NO for . . . no)

Ok, suRe, yup

(NOTE: The dictionary is a disk file, LarrysYES\_Dictionary, that consists of yes, YES as well as all items added during all sessions of “A”)

<<< Hit ENTER to continue >>>

MENU

S – Sign in

L – Display Logo

A – Start Affirmative Interaction

D – Display Dictionary

X – Exit Program

Please enter your selection and hit RETURN: x

Thank you for selecting “x” . . .

Time to x-cape!

Farewell, Albert, and thanks for visiting Larry’s Affirmative Interaction Program

Display the current date and time

Display your complete ID INFORMATION (nicely formatted),

CREDITS and any STARS you did in the following format (BTW – Credit given only for STARS that are completed and fully displayed here):

STARS

1. Fancy logo
2. Early demo
3. & 4. PowerPoint presentation

about artificial intelligence

TOTAL STARS = 4

<<< Hit ENTER to continue >>>

**BTW: No “double-dipping” from previous assignments on the STARS**

**STAR: Allow only one sign-in. Alert user it can’t be done if user tries again**

**STAR: Only proceed after sign-in is done, although it’s okay to exit**

**STAR: Only proceed if sign-in is done first, although it’s okay to exit**

**STAR: Allow only 3 invalid menu entries after which program “exits”**

**STAR: Expand previous STAR to warn user at 3rd invalid entry that program will exit if the 4th try is invalid.**

**STAR: Instead of re-writing the menu for “Sorry” message, be “ecological “and move the cursor back to the prompt point, but be sure to pause for a moment and then erase the now unnecessary “Sorry” part of the message**

**STAR: Expand previous STAR to be ecological for all appropriate menu selections**

**5** – **SAVE** your file early and often -- like every 5 minutes. And, use your backup "disk"

**6** – **TEST** your file early and often -- like every 5 minutes -- How do you eat an apple?

**7** – **PROGRAM DOCUMENTATION**

1. First, include via comments your complete ID INFORMATION, PROGRAM DESCRIPTION, **CUSTOM-DEFINED FUNCTION LIST** and CREDITS (to those who helped you and whom you helped).
2. **Include "inline credits" to acknowledge specifically where you were helped.**
3. Add comments immediately before each segment of your program to describe "highlights" of coming attractions. Insert at least one blank line before each of these comments. Make all function, variable and const names self-descriptive, clear and fully formed (no abbreviations or secret code names). **Use verbs for function names, like “displayXyz” or “getXyz” or “calculateXyz” or “doXyz” and nouns for constants and variables.**
4. Define a const for your name and use it whenever your name appears
5. Add “banners” for: your prototypes to include description of each function, all constants, variables, start of function definitions, end of function definitions and in the function definitions with a description of each function (you can use the same ones as in your prototypes). See Handout #1.2 for all the details.
6. **FUNCTION PROTOTYPES – In your function descriptions, be sure to include how the parameters are used and for non-void functions identify what information will be returned. Make sure to say “return x” for non-void functions. For example, consider this hypothetical function prototype:**

**//NAME: getFavoriteNumber**

**//DESCRIPTION: Prompts user by name for a favorite number and**

**returns it**

**int getFavoriteNumber( string userName )**

1. IMPORTANT: Remember to embrace the basic Michelangelo structure for **every** C++ program.

**8** – **DEMO (= beta testing)** your program in the Lab with a completed TASK SHEET

**9** – **HAND IN HARDCOPY** of your accurate TIME SHEET

**10** – **STARS (One STAR for each item, but no “double-dipping”)**

1. Work in a 2-3 person team to write and demo one program together that contains all the required information. Additionally, include the names of all the team members (as separate constants) in the welcome and farewell AND include each team member’s individual logo when displaying the logos. Add additional and different cellular automata patterns, one for each teammate. In the ID INFO, individually show hours and difficulty for each person. **Also, in each function, document the names of the contributors and the percentage each one contributed to the function (1 STAR per teammate).**
2. In a NEW engaging way, use >=4 windows.h colors throughout
3. Implement this assignment by creating and using a custom header file to hold all your function prototypes and definitions. Include your Michelangelo documentation with ID INFO, etc. in the header file
4. Add extra pizzazz and briefly explain what you did:
5. Effectively use three new "Advanced" features (= not yet introduced). List them:
6. Demo before the due-date (N.B.: You still can do more STARS on due-date)

**“Imagination is the beginning of creation. You imagine what you desire, you will what you imagine, and at last, you create what you will”**

**– George Bernard Shaw**

**“Originality is nothing but judicious imitation”**

**– Voltaire**

