3. Written Responses (Created Independently) 2021

Submit your responses to prompts 3a – 3d, which are described below. Your response to all prompts combined must not exceed 750 words (program code is not included in the word count). Collaboration is **not** allowed on the written responses. Instructions for submitting your written responses are available on the AP Computer Science Principles Exam Page - you will need to copy and paste each separate response into the corresponding box on the AP Digital Portfolio.

a. Provide a written response that does all three of the following:

Approx. 150 words (for all subparts of 3a combined)

i. Describes the overall purpose of the program

The purpose of the program is to connect users in real time over a website. This website has a client webpage (In JavaScript, HTML, and CSS) that the users connect to and a server program (In Python) that handles the transfer of messages.

ii. Describes what functionality of the program is demonstrated in the video

The video shows two users connecting the website, sending messages, and receiving the other users' messages in real time.

iii. Describes the input and output of the program demonstrated in the video

The program input is the users submitting their message on the page. The program outputs when it displays the new messages on the webpage.

b. Capture and paste two program code segments you developed during the administration of this task that contain a list (or other collection type) being used to manage complexity in your program.

Approx. 200 words (for all subparts of 3b combined, exclusive of program code)

i. The first program code segment screenshot must show how data have been stored in the list.

```python
# Instantiate list of messages
messages = list()

# Socket input event
# Message comes in from client (the parameter message)
# Parameter message is a dictionary containing username, message, and time

@socketio.event
def updateMessage(message):

    # Time is converted from UNIX timestamp epoch into human readable format
    message['time'] = datetime.utcfromtimestamp(int(message['time']) / 1000)
    message['time'] = message['time'].strftime('%m / %d<br>%Y<br>%H:%M:%S')

    # Calls makeSafe procedure to prevent users from rendering their own HTML
    message['username'] = makeSafe(message['username'])
    message['msg'] = makeSafe(message['msg'])

    # Appends message to list of messages
    messages.append(message)

    # List of messages is sent to all the clients to be updated
    emit('newMessage', messages, broadcast=True)
```

ii. The second program code segment screenshot must show the data in the same list being used, such as creating new data from the existing data or accessing multiple elements in the list, as part of fulfilling the program's purpose.

```
/*
* This procedure shows the list being used
*
* Procedure for displaying messages
* Takes paramter of message array from server
* Iterates through messages and renders them
* on the HTML page
*/

function displayMessages(servedMessages)
{
    // Get messages as array from server input
    var messages = Array.from(servedMessages);

    var table = '';
    // For each message in messages
    for (var i=0; i < messages.length; i++)
    {
        // Adds HTML for each message
        message = messages[i];
        table += `<tr><td class="username">${message.username}</td>`;
        table += `<td class="msg">${message.msg}</td>`;
        table += `<td class="time">${message.time}</td></tr>`;
    }

    table += '</table>';
    // Render table in document
    document.getElementById('messages').innerHTML = table;
}
```

Then, provide a written response that does all three of the following:

iii. Identifies the name of the list being used in this response

messages

iv. Describes what the data contained in the list represent in your program

The list “messages” contains the messages that users have sent. Each message element in the list is a dictionary with the attributes username, msg, and time. These respectively hold the username of the user, the message they sent, and the time it was sent at.

v. Explains how the selected list manages complexity in your program code by explaining why your program code could not be written, or how it would be written differently, if you did not use the list

The list manages complexity because the number of messages posted by users grows and gets very large. It could not be written without a list because there would need to be a separate variable for every message, and that cannot be done during runtime, which is when messages are being added.

c. Capture and paste two program code segments you developed during the administration of this task that contain a student-developed procedure that

implements an algorithm used in your program and a call to that procedure.
Approx. 200 words (for all subparts of 3c combined, exclusive of program code)

i. The first program code segment screenshot must be a student-developed procedure that:
   - Defines the procedure's name and return type (if necessary)
   - Contains and uses one or more parameters that have an effect on the functionality of the procedure
   - Implements an algorithm that includes sequencing, selection, and iteration

```
# Procedure makeSafe
# Parameter unsafeText is the text from the user
# Returns safeText, the safe unsafeText
# Iterates through string of text to search and replace characters
# <  : &lt;
# >  : &gt;
# \n : <br>

def makeSafe(unsafeText):
    safeText = str()

    # Sequencing and iteration (through string)
    for i in range(len(unsafeText)):
        char = unsafeText[i]

        # Selection
        if char == '<':
            safeText += '&lt;'
        elif char == '>':
            safeText += '&gt;'
        elif char == '\n':
            safeText += '<br>'
        else:
            safeText += char

    return safeText
```

ii. The second program code segment screenshot must show where your student-developed procedure is being called in your program.

```
# Calls makeSafe procedure to prevent users from rendering their own HTML
message['username'] = makeSafe(message['username'])
message['msg'] = makeSafe(message['msg'])
```

Then, provide a written response that does both of the following:

iii. Describes in general what the identified procedure does and how it contributes to the overall functionality of the program

The procedure takes text and converts certain characters from a piece of text to versions that are safe to render on the HTML page. This contributes to the security of the program, as it prevents users from cross-site scripting. It also converts “\n” to “
”, which allows carriage returns to be rendered on the page correctly.

iv. Explains in detailed steps how the algorithm implemented in the identified procedure works. Your explanation must be detailed enough for someone else to recreate it.

The algorithm takes a string parameter, which is a composite data type of a list of characters. The program iterates through this string parameter, which is the unsafe text, and looks for the characters to be removed. If the character is HTML safe, append it onto the new return string. If not, append the HTML safe version of the character to the new string, rather than the original character.

d. Provide a written response that does all three of the following:
Approx. 200 words (for all subparts of 3d combined)

i. Describes two calls to the procedure identified in written response 3c. Each call must pass a different argument(s) that causes a different segment of code in the algorithm to execute.
First call:

```
message['username'] = makeSafe(message['username'])
```

Second call:

```
message['msg'] = makeSafe(message['msg'])
```

ii. Describes what condition(s) is being tested by each call to the procedure
Condition(s) tested by the first call:

The first call passes the username as a parameter. This input can be different every call, as it operates on the username that the user specified in their post.

Condition(s) tested by the second call:

The second call passes the user's message as a parameter. This input can also be different every call, as it depends on the contents of the message that the user wrote in their post.

iii. Identifies the result of each call
Result of the first call:

The result of the first call is the user's username with HTML safe characters.

Result of the second call:

The result of the second call is the user’s message with HTML safe characters.