Create – Applications From Ideas
Written Response Submission Template

Please see Assessment Overview and Performance Task Directions for Student for the task directions and recommended word counts.

Program Purpose and Development

2a)

I made Classroom Name Caller (CNC), a student picker software in JavaScript. CNC's purpose is to help teachers pick students randomly. CNC first let's teachers add students to the list, then displaying those students. Afterward, the teachers can pick one, two, or partner all the students up according to the button pressed, with the results shown in the div tags. Finally, the reset button clears the div tags, the text in the text box, and the current students, and the instructions button toggles the instructions. This video illustrates the above by showing the chosen students being added to the text box, and then the buttons for choosing one, two, and partnering the students up being clicked and the results showing in the div tags. It finally shows the instructions toggling visibility by clicking the button for it, and then the information for the page being cleared with the reset button.

2b)

An incremental process that resolved difficulty was creating the pick2() and pick3() functions, which choose partners together. At first, I tried to splice the student array in half, and give each half to either function, but the original studentArray would have been compromised. This was fixed by finding the length of half of the studentArray(rounded) and assigning one half of the studentArray to each function. Another incremental process incorporated into the code is a for loop that groups different people depending on their indexes in the studentArray. Because the same groups of people would repeat, I did some research and found how to shuffle the items in my array of partners, effectively changing their indexes. This allowed me to call on the same indexes while changing the item assigned to the index. An iterative process is my debugging process, which helped me fix problems with my code. If a problem aroused, I comment different sections of my code that relates to my problem. This allows me to find my problem, as the code will work fine only after I comment on the bad part of my code.

2c)

```
function partner()
{
    var partnerArray = [];
    var arrayLength = studentArray.length

    if(arrayLength % 2 == 1 )
            {

                studentArray.push("no one else")

            }

    for(var i = 0; i < arrayLength; i += 2)
    {
        var partner1 = studentArray[i]
        var partner2 = studentArray[i + 1]

        console.log(partner1)
        console.log(partner2)
        partnerArray.push(partner1 + " and " + partner2);


    }
    console.log(partnerArray)
    display(partnerArray)

    if(studentArray[arrayLength] = "no one else")
        {
            studentArray.pop()
        }

    shuffle(studentArray)
}
```

An example of an algorithm in my program is the function partner(). partner() is used when the user presses the partner up button after adding students to the studentArray. First, the function generates an empty, local array called partnerArray. It then finds the length of studentArray and checks if the length is a multiple of 2 with modular division. If it isn't, it adds the string "no one else" to studentArray. A for loop is then run, which says that while the variable i, which starts at zero, is less than the number of students, name the person at the position i in the studentArray as partner 1, and the person at position i+1 in student array as partner two. Push the final result, partner1 + "and" + partner2, to the partnerArray, then add i+2 and repeat. When the for loop is finished, we

check if the last value in the studentArray is "one else". If it is, the program removes it. When finished, it runs the shuffle function to assign new indexes to the students in studentArray. This allows for the function to run again, and display new partners, as new partners will be at index i.

2d)

```
function pick3()
{
arrayLength = studentArray.length
var halfArray = Math.round(arrayLength/2)
console.log(halfArray)
var arrayPerson = randomInt(0, halfArray - 1)
var personPicker = studentArray[arrayPerson];
partnerArray.push(personPicker)
console.log(personPicker)

}

'*Description: Picks a random student from the 2nd half of studentArray and ad
*
*Parameters: Needs to have at least one students in the 2nd half of the stude
*
*Returns: A random person for the 2nd half of the studentArray is added to pa
*
*/


function pick2()
{
arrayLength = studentArray.length
var halfArray = Math.round(arrayLength/2)
console.log(halfArray)
var arrayPerson = randomInt(halfArray, arrayLength-1)
var personPicker = studentArray[arrayPerson];
partnerArray.push(personPicker)
console.log(personPicker)

}


'*Description: Picks twos students from the studentsTyped in
*
*Parameters: Needs the button to be clicked while two students are in the lis
*
*Returns: Two random students, who aren't the same person, displayed in a div
*
*/

function pair()
{
idDivOutput.innerHTML="";
partnerArray = [];
console.log(partnerArray[0])
console.log(partnerArray[1])
pick3();
pick2();
display(partnerArray[0] + " and " + partnerArray[1] + " have been chosen!")

}
```

An abstraction programmed in CNC is pick2() and pick3() , which is used in the function pair(). After the user has added the list of people(in the form of studentArray) to choose from using addPeople(), pick2() and pick3() choose those people from different halves of the list, and displays them using pair(). An example of a mathematical concept is dividing the length of studentArray by 2 to find the middle index, and choosing students whose indexes are less or more/equal with pick2() and pick3() respectively. An example of a logical concept used is replacing the old partners with new partners in student array. My function shows this by showing new partners despite calling on the same index of partnerArray. This managed the complexity of the program replacing information rather than adding new information and calling new indexes. Because pick3() and pick2() choose the people added to the array, pair() uses less programming, and debugging is made easier due to the code being split up, which helps in isolating problems. I did this when debugging pair(), and found out that pick3() wasn't choosing specific people, and was able to fix it by changing the parameters on who it can pick.