ASEE's 123rd Annual • Conference & Exposition • New Orleans, LA • June 26-29, 2016

Paper ID #15630

Innovative Activities to Teach Computer Science Concepts Inside the Classroom and at Outreach Events

Dr. Stephany Coffman-Wolph, West Virginia University Institute of Technology

Dr. Stephany Coffman-Wolph is an Assistant Professor at West Virginia University Institute of Technology (WVU Tech) in the Department of Computer Science and Information Systems. She is a founding member of AWESOME (Association for Women Engineers, Scientists, Or Mathematician Empowerment) at WVU Tech and currently serves as a co-Advisor of the student organization. Other research interests include: Artificial Intelligence, Fuzzy Logic, Game Theory, and Software Engineering.

Innovative Activities to Teach Computer Science Concepts Inside the Classroom and at Outreach Events

Teaching an introductory course in computer programming can be challenging. Additionally, introducing grade school, middle school, and high school students to computer science without a computer lab seems impossible. The activities presented in this paper do not require a computer lab and can be done with a range of age groups, any number of people, and people with no prior computer experience. The goal is to introduce various topics using fun physical activities and everyday experiences that are familiar. Most young kids and even some college students seem to be unaware of what computer science means. These activities help them understand the depth and diversity a computer science undergraduate degree can entail.

The paper will provide the details for each of these activities and the learning objectives. The activities included are:
(1) Binary, Octal, and Hexadecimal Initial Keychains
(2) Understanding Variables and Arrays with Paper Bags
(3) Branching and Looping Statements with Starburst Candies
(4) General Class Structure with Bags, Boxes, and a Bin
(5) Dr. Doolittle's Vet Office: Learning Classes with Stuffed Animals
(6) Arrays with Tissue Boxes, DVD Sets, Paper Plates, and other Household Goods
(7) Basic Networking, Message Passing, and Security with Party Hats and Candy
(8) Monster's Hate Chocolate: Learning Try/Catch Blocks

Introduction and Background

Faculty members of West Virginia University Institute of Technology (WVU Tech) are encouraged to participate in the ongoing outreach activities on and off campus. Additionally, as a female faculty member in a STEM field, the author is acutely aware of the need for female role models and has a keen interested in participating in outreach activities geared towards encouraging young women's interest in STEM.

As a faculty advisor and founding member of the Association of Women Engineers, Scientists, Or Mathematician Empowerment (AWESOME) at WVU Tech, a faculty and student organization dedicated to recruiting and retaining future generations of females into the STEM fields, there have been many opportunities to participate in outreach and great encouragement to develop activities to specifically promote computer science. This led to the development of fun, interactive, easily transportable, and low cost activities that could be used to introduce a wide range of ages to computer science – from young kids (e.g. Daisy Scouts/Cub Scouts) to teenagers (e.g. Juniors and Seniors in High School). Generally, access to a computer lab is impractical or just not available (and not easily transportable) and, thus, these activities were all designed to be done without computers, to go beyond simply teaching someone to program and to focus on the concepts behind programming and the "science" of computer science. The author would argue that most careers in computer science go beyond simply sitting around programming and require many other skills. The activities outlined below provide an introduction to some of these other required skills. Additionally, all activities are either programming language independent or

could be tailored to whatever specific programming language you wish (or be done using generic pseudo-code).

Two of the activities presented here, the Binary, Octal, and Hexadecimal Initial Keychains and Basic Networking, Message Passing, and Security with Party Hats and Candy, were inspired by the Computer Science (CS) Unplugged website [1, 2]. CS Unplugged [1, 2] provided videos, worksheets, and teaching guides to a variety of computer science activities. All other computer science activities presented in this paper are the creation of the author. In all cases, the activities in CS Unplugged [1], [2], and here, are designed to be done without a computer. All of these activities [1, 2] are general computer science concepts and not computer programming concepts. This paper includes several general computer science concept activities, as well as presenting the details for activities focused on general programming concepts. Other papers [3, 4] demonstrate that hands-on activities on computer science topics increase student awareness of the field of computer science and interest in the field.

Figure 1:
Binary, Octal, and Hexadecimal Initial Keychains

Materials:
- Pony Beads in 2 colors (binary), 8 colors (octal), and 16 colors (hexadecimal)
- 3 pieces of Cord cut at approximately 10 inches for each participant
- 1 Keyring for each participant
- ASCII Table(s) showing binary, octal, and hexadecimal

Prep:
- Purchase pony beads
- Place each color of bead in a separate container with a tag or sign to designate the number it represents. Using containers with easily removable and replaceable lids makes set up and clean up significantly faster
- Print ASCII Tables

Activity Instructions:
1. Pass out to each student the 3 cord lengths and a keyring. (You can either pre-tie the cord lengths to the keyring or have the children do it at this time).
2. Start with 2 colors for the binary beads to represent 0 and 1. The student's should find the first letter in their first name on the ASCII table and string that sequence of beads onto the cord. After they have finished the first initial, they should do their middle initial and, finally, their last initial. Tie the end to keep the beads from slipping
3. Swap out the 2 color binary beads with the 8 colors for octal. Repeat the process.
4. Swap out the 8 colors for octal with the 16 colors for hexadecimal. Repeat the process.

Suggestions for Colors:
- For the binary activity pick two very different colors (white and black, yellow and blue, red and black). Possibly consider using your school or university colors.
- For octal and hexadecimal it is best to have 8 or 16 distinctly different colors. Otherwise, kids can get confused between the colors. However, it is completely fine to use a light and dark version of the same color (e.g., light blue and dark blue)
- Using the same colors for the numbers that are the same (i.e., always using the same color for 0) can make it easier to move from binary to octal to hexadecimal
- For octal or hexadecimal, a possible color selection is 8-color and 16-color schemes for older monitors.

General Suggestions for the Activity:
- Providing a "guide" sheet (a spare piece of paper to place under the current line on the ASCII table)
- For all day events at a table, it is easier to place your ASCII table sheets in plastic sheet protectors – they will last longer, not get as bent, and be more water/hand sanitizer resistant
- For the ASCII table, consider providing the directions in multiple formats (e.g., binary, color squares, and the words written out) – this will make the activity more accessible to students
- Younger students find tying the cord difficult to get tight enough, so extra help might be needed

Alternatives:
- For a quicker activity or when working with younger students, skipping the octal portion and focusing on binary and hexadecimal
- You can make bracelets instead of keychains. Use the same cord and tie it like an old-fashion friendship bracelet (i.e., making it slightly adjustable). An alternative would be to use stretch cord or elastic
- For most age groups, we have provided only capital letters from the ASCII table. However, for older students including uppercase, lowercase, and punctuation in ASCII for the three formats can enhancing the activity. (You will need to provide cording of longer than 10 inches).

Binary, Octal, and Hexadecimal Initial Keychains (or Bracelets)

This activity's primary goal is to demonstrate that computers "speak" in the language of numbers and not the way humans do. In this day and age, most kids have interacted with a computer at some point – even very young kids. Additionally, with the popularity of Minecraft, students are slightly more familiar with some of these concepts. This activity has been used successfully with a wide range of age groups and in various settings (booth/walk up table or school/classroom

visit). Participants create keychains or bracelets with their initials in binary, octal, and hexadecimal. This activity was inspired by an activity in CS Unplugged [2]. For detailed instructions and suggested variations, please see figure 1.

Learning Objectives: After this activity, kids should be able to:

- Explain that computers "speak" a different language than humans (all ages)
- Understand that using Octal or Hexadecimal is shorter than binary (all ages)
- Understand that each letter is represented by the same length and, therefore, do not have "spacers" (all ages)
- Understand that the color is just representative of numbers and can be done with any set of colors as long as you have the correct number (middle school and up)
- Define the terms bit, byte, and nibble (middle school and up)
- Understand that ASCII is a conversion between keyboard characters and computer "speak" (middle school and up)
- Understand that ASCII represents all of the keys on the keyboard (middle school and up)
- Understand what ASCII (American Standard Code for Information Interchange) means and represents (high school and up)
- Understand the mathematical relationship between binary, octal, and hexadecimal (high school and up)

Understanding Variables and Arrays with Paper Bags

Introducing the concept of variables and the various types (Boolean, character, integer, double, string) is a basic understanding needed for any programming language. Although some students are familiar with the concept of a variable in mathematics, students find the programming version to be very confusing. Some examples: many students are befuddled by the idea of multiple different data types and that the equal sign (=) denotes assignment and does not represent equality. It has also been the author's experience that students illogically think that a variable records previous values and they often do not understand that a variable can be null. Expanding the basic variable to the concept of an array can be equally challenging for students. The fact that a single identifier can represent multiple items, each "item" in the array can be accessed individually, and each have a different value is not intuitive for the majority of students. This activity, detailed in Figure 2, makes use of commonly found paper bags to discuss variables and arrays.

Learning Objectives: After this activity, students should be able to:

- Define, in their own words, a variable (all ages)
- Define, in their own words, an array (all ages)
- Explain the difference between a single variable and an array (all ages)
- Understand that there are differences between variable sizes (all ages)
- Explain the purpose of a variable in computer programming (middle school and up)
- Understand the correspondence between the visual bag demonstration and provided programming code (middle school and up)
- Understand that "=" means assignment in programming (middle school and up)
- Explain the different bit/bytes each variable will hold (high school and up)

- Create the matching programming code from the visual bag demonstration (high school and up)
- Understand that most programming languages start indexing for arrays at 0 (high school and up)

Figure 2:

Understanding Variables and Arrays with Paper Bags

Materials:

- Paper bags in 5 different sizes (and several of each size)
- Stapler and/or Masking tape
- Thick Permanent Marker in a Dark color
- White paper and (optional) Large sticky notes
- Chalkboard, Whiteboard, Computer connected to an overhead projector, or Large paper flip-pad (used to show corresponding code as you do the activity)

Prep:

- Start with the smallest paper bags you have – label it with Boolean. Label the other bags (in increasing size order) character, integer, double, and string. (Use the naming convention of the programming language of your choice)
- Prep pieces of paper (of corresponding sizes that will fit in your paper bags) with various "values"
- Prep pieces of paper or sticky notes to "name" the various paper bags as you would variables. (Use naming conventions to match your class or personal choice)

Activity Instructions:

1. Set up one of each type of bag (label away from the students) from smallest to largest in the front of the classroom. Ask and discuss with the students what is different about the bags and what is similar. (Goal: to have them realize that the bags go in increasing order and that there are 5 different sizes)
2. One at a time, turn the bag (label towards the students) from smallest to largest. Discuss the various data types – what can be stored in the variable, size, etc.
3. Use your prepared paper or sticky notes to name the first variable. Show the corresponding programming code on the chalkboard (or display device of your choosing)
4. Use your prepared paper to assign a variable a value. Display the corresponding programming code. Use a different prepared paper to assign a variable a new value – remember to make a big show of taking the old value out of the bag first (because only one value can be assigned to a variable at a time) and replace it with the new value. Display the corresponding programming code
5. Repeated this process with all the variable types you wish to demo
6. Select one of the variables types (suggestion: int). Set up multiple bags of the same size in a row. Use a stapler or tape to attach the bags together in a row. Ask and discuss with students the difference between the single bag and the row of multiple bags. (Goal: to have them realize there are multiple values being grouped together vs. a single value)
7. Explain and discuss the array and the use in programming languages. Give an example of why you might want multiple values grouped together.
8. Use your prepared paper or sticky note to name the array. Display the corresponding programming code
9. Use your prepared paper to assign a value to one of the elements in the array. Demonstrate/discuss how indexing is done in computer sciences and discuss how you start counting at 0. Display the corresponding programming code. Continue assigning variables until the entire array is filled with values

Suggestions:

- For Boolean variables, use paper sandwich sleeves as they do not hold much and illustrate the size of true or false. For integer, use a lunch sack as they are inexpensive and easy to staple together for the array portion of the activity. For double, use paper grocery bags and used handled shopping bags for string. For char, use a "mini" lunch sack size
- If you wish to expand the types of variables in the demonstration, simply find more sizes of bags
- For large groups, "pre-name" all the variables. For small groups, let the students select or make-up names
- For younger students or for short time periods, skip the array portion of the activity or do a smaller number of variables types. Minimally you can demonstrate the concept of size difference with two types of variables

Branching and Looping Statements with Starburst Candies

This hands-on activity demonstrates the concept of branching and looping statements found in almost all programming languages. Additional skills that are important to future computer science students is teamwork and reviewing/discussing code (often known as code review). By completing the provided instruction sheets (Figure 4 and 5), students encounter branch statements and looping. Instruction sheet Part #1 contains a series of statements for the group to determine who will act as team lead (similar to a series of if statements). After completing the first instruction sheet, the group receives a bag of Starburst candies and a second instruction sheet that uses the entire group to simulated a flag controlled while loop. Afterwards, each group creates flowcharts (and code) for each part of the activity. The details for this activity are found in Figure 3.

Figure 3:

Branching and Looping Statements with Starburst Candies

Materials:

- Starburst candies
- A solid color bag (suggestion: paper lunch bags)
- For prep: printer and copier to create instruction sheets
- White paper (minimally 1 per group, but easier to give one to each student)
- Optional: Chalkboard, Whiteboard, Computer connected to an overhead projector, or Large paper flip-pad (used to show flowcharts and code as you do the activity)

Prep:

- Print out and/or copy the 2 instruction sheets (Figure 4 and Figure 5)
- Prepare a number of bags with an assortment of Starburst candies in a variety of colors, except only have 1 red colored Starburst candy.
- Print out and/or copy the Skeleton pseudo-code worksheet

Activity Instructions:

1. Have students get into groups of 3-5
2. Pass out the first of the instruction sheets (part 1). Have the students begin part 1 – which will select a team lead and set up for part 2. When the students finish part 1, it will instruct the group to send the team lead to show you part 1
3. For each team lead that shows you a correct part 1, give them a part 2 instruction sheet and one prepared bag. Have the students begin part 2
4. After completing the second part, briefly discuss the actions that were done with the two instruction sheets. Introduce the basics of flowcharts (decisions, actions, arrows to indicate flow). Pass out blank sheets of paper. Have the groups create a flowchart for part #1 and part #2
5. Show possible solution(s) for the flowcharts. Ask and discuss with students how their solutions are different
6. (Optional) Pass out more blank paper (if needed). Have the students write programming code or pseudo-code based on the flowcharts

Suggestions:

- Instead of Starburst candies, you could use M&Ms or Skittles or any other candy that has multiple colors (minimally 2 colors are needed)
- For really younger students, you may wish to skip the flowcharts and coding
- For middle school students, you may need to "guide" them more when translating between the flowchart and programming code. One option is to provide skeleton pseudo-code where they only have them fill in essential elements

Figure 4: Instruction Sheet Part 1

Part #1:

Please follow the directions as stated. Make sure you do each step and read carefully.

1. If the team has an even number of people, then write down each team member's first name and birth month (as a number)
   Else, write down each team member's first name and day of birth.
2. If the team has an odd number of people, re-write the list in ascending order by day of birth.
   Else, write the list in descending order by birth month.
   In either case: In the event of ties, order by who was born furthest from our current location.
3. Select the first name off the list to be team leader of the group today.
4. Double check you followed the directions above correctly.

Have the team leader bring your list to the instructor to receive the next part of the activity

Figure 5: Instruction Sheet Part 2

Part #2:

Please follow the directions as stated. Make sure you read carefully.

Step #1: Using your list generated earlier, begin with the last person on the list. This person is designated as "current". Hand "current" the bag received from the instructor

Step #2: Have the "current" person draw 1 item from the bag

Step #3: If the item's color == Red, Go to Step #4

Else, Go to the next person "back" in the list, this person will become the new "current". If you are at the beginning of the list, return to the last person in the list. (i.e., you should be following the list in reverse order and repeating the list until the item's color is red). Go to Step #2

Step #4: Answer the following questions:

1. How many passes did you make of step #2?
2. How is this number related to the number of items that had to be drawn before the red item was found?

Learning Objectives: After this activity, students should be able to:

- Define, in their own words, branching and looping (all ages)
- Explain the basics of it then else statements (all ages)
- Explain how looping is repeating a set of actions multiple times (all ages)
- Practice group work and teamwork skills (all ages)
- Explain how looping and branching are used in programming languages (middle school and up)
- Define, in their own words, a flowchart (middle school and up)
- Explain the different ways to control a loop (high school and up)
- Explain the purpose of a flowchart in relationship to programming (high school and up)
- Define, in their own words, pseudo-code (high school and up)
- Create pseudo-code/programming code given a flowchart (high school and up)

General Class Structure with Bags, Boxes, and a Bin

This activity is designed to introduce and provide students with a visual reference for the common programming concept of a Class. This simple demonstration, detailed in Figure 6, builds on a previously mentioned activity – "Understanding Variables and Arrays with Paper Bags" (and also reuses the bags from this activity). This activity assumes that students are already familiar with both the concept of variables and methods/functions.

Figure 6:

General Class Structure with Bags, Boxes, and a Bin

Materials:

- A large plastic bin (with lid)
- Several small boxes or bins (various sizes) that can easily all fit inside the plastic bin at the same time
- Paper bags from the "Understanding Variables and Arrays with Paper Bags"
- Optional: Chalkboard, Whiteboard, Computer connected to an overhead projector, or Large paper flip-pad (used to show code as you do the activity)

Prep:

- Decide on the Class name and functionality you wish to demonstrate. Write the corresponding programming code for that class
- Create a label for the large bin with the class name
- Locate or recreate variable bags from the "Understanding Variables and Arrays with Paper Bags" and create labels to name the data members
- Make sure all components (bags, boxes) will easily fit within the plastic bin (at the same time) and you can close the lid
- Create labels for the boxes/bins (i.e., the various methods/functions for the class)

Activity Instructions:

1. Begin by placing the large empty bin at the front of the classroom. Introduce the concept of a class to the students
2. Label the large empty bin the class you are going to create. Show the corresponding programming code to the students
3. Introduce the concept of data members. Using the variables bags, add data members to the class by placing the variable bags into the large bin. Show the corresponding programming code to the students
4. Add methods to the class by labeling a smaller box/bin and placing it into the large bin. Show the corresponding programming code to the students as you go along
5. Place the lid on the class. Discuss how the plastic bin keeps all the parts together and as a group. Discuss the complete code (i.e., anything between the {} is inside the bin).
6. (Optional) Discuss how the class is like a blueprint and a data type. Demonstrate the code for creating an instance of a class with specific values. You can add values into the bags (as was done with the "Understanding Variables and Arrays with Paper Bags"). Additionally, you can add values into bags and demonstrate passing the values to the methods/functions as arguments/parameters.

Suggestions:

- When selecting a Class to demonstrate, I like to pick something students are familiar with and is a little humorous (Superhero, Cookies, Pizza, Monkey, Snowman)
- If working with older students or smaller groups, I often let them select/decide/suggest the data members and methods for the class

Learning Objectives: After this activity, students should be able to:

- Define, in their own words, a class (all ages)

- Define, in their own words, data members (all ages)
- Explain how each of the boxes represents one of the various methods/functions written for the class (middle school and up)
- Explain how the Class acts as a blueprint for a data type (high school and up)

Dr. Doolittle's Vet Office: Learning Classes with Stuffed Animals

This activity, like the previous, focuses on the important object-oriented programming concept of Classes. However, this activity focuses on designing a Class while the previous activity focused on the mechanics of creating a specific Class and syntax of a Class. In this activity students are designing and creating a Class from a verbal or written description – thus further expanding their knowledge as well as understanding how a computer scientist goes about translating from "English" to code. Knowing the syntax of Classes is important, however, computer scientists also need to be able to make design decisions and understand the good and bad of those decisions. This activity assumes that students are already familiar with the syntax and basics of a Class. The details for conducting the activity are provided in Figure 7.

Figure 7:

Dr. Doolittle's Vet Office: Learning Classes with Stuffed Animals

Materials:

- An assortment of stuffed animals or animal figurines (or at an absolute minimum pictures of animals)
- A table, desk, or area to set up the animals so students can easily see them while creating and designing the class
- Blank sheets of paper
- Chalkboard, Whiteboard, Computer connected to an overhead projector, or Large paper flip-pad (used to show code as you do the activity)

Prep:

- Prepare stuffed animals or animal figurines – they can be any animals (real or imaginary). Recommend at least 3 or 4

Activity Instructions:

1. Begin by setting up the stuffed animals one-by-one for the students to see. Ask and discuss the similarities or differences between the various animals. Make a short list of the similar characteristics.
2. Tell the students that Dr. Doolittle wants a computer program for his veterinary office. In order to write the program efficiently, you are first going to create a class called Animal.
3. Pass out the blank paper. Break the students up into groups of 3 or 4, have them create a list of important characteristics or information a vet might need on every animal. Tell them to focus on thinking up things that are true for the majority of animals (not specialized characteristics)
4. Discuss the characteristics each group has come up with and generate a master list
5. If needed, pass out additional blank paper. Have the groups begin to design the class Animal. Focusing on only the variables using the master list
6. Discuss the variables each group has come up with and use it to generate a possible solution
7. Ask the groups to discuss and decide on 3-5 methods/functions for the Class. (Optional) If the programming language you are studying has accessors and mutators, have the students also decide which variables should be accessed and which variables should be allowed to be changed.
8. Discuss the methods/functions each group has come up with. (Optional) Discuss the accessors/mutators that each group came up with. Update the solution
9. Using the stuffed animals or animal figurines, have the students create specific instances of class Animal. (Optional if time) Use one or two methods from the class to change values or "do" something from a verbal or written description

Suggestions:

- For younger students, you might want to skip the accessor/mutator portion
- For older students or small number of students, you can have the students "lead" more of the design. For a large number of student, you might want to prep a solution and slowly reveal it.

Alternatives:

- You can use this activity to talk about inheritance. You could have Animal be the base class with two derived classes Pet and Zoo. Ask students to identify characteristics specific to Pet vs. Zoo and what characteristics are the same (and, thus, should be in the base class)

Learning Objectives: After this activity, students should be able to:

- Explain, in their own words, Class and data members (all ages)
- Work effectively within a team environment (all ages)
- Explain and justify design decisions with others (high school and up)

Arrays with Tissue Boxes, DVD Sets, Paper Plates, and other Household Goods

This demonstration, detailed in Figure 8, is great for introducing the general concepts of arrays (or other similar data structures) using common household items. Unlike like the one-dimensional array concept introduced with the bags, this activity will easily allow one to also introduce the concept of two-dimensional arrays. The idea behind this demonstration is to help students identify 1D and 2D array-like structures in everyday life. This will help students form a mental picture of an array and, thus, improve the understanding of the use of arrays in programming.

Learning Objectives: After this activity, students should be able to:

- Define, in their own words, one-dimensional and two-dimensional arrays (all ages)
- Explain the similarities and differences between one-dimensional and two-dimensional arrays (all ages)
- Provide an example of when a 1D or 2D array could be used in programming (high school and up)

Figure 8:

Arrays with Tissue Boxes, DVD Sets, Paper Plates, and other Household Goods

Suggested Materials:

- Multi-packs of facial tissue in a variety of sizes
- DVD box sets (especially the multiple boxes in the sleeves and if you have multiple seasons of the same TV show)
- Package of paper plates, napkins, plastic cups
- Package of cookies that are stacked on end
- Box of candy with the individual places for each piece (empty or full)
- Empty plastic storage box with the dividers (found at craft or hardware stores)
- Stack of all same size Legos or Mega Blocks
- Any food or household item that comes in a multi-pack – especially anything that resembles a table or grid
- Sticky notes or pieces of paper cut into approximately 3 inch by 3 inch size
- Black (thick) marker
- Chalkboard, Whiteboard, Computer connected to an overhead projector, or Large paper flip-pad (used to show code as you do the activity)

Prep:

- Purchase, find, or borrow a wide variety of the suggest materials listed above
- (Optional) Prepare sticky notes or the 3X3 sheets of paper with numbers, characters, or strings

Activity Instructions:

1. Display your various examples. Hold each up and discuss what it is.
2. Have the student brainstorm on the similar characteristic(s) of all of the items
3. Discuss the students' brainstorm ideas – and keep discussing until each of the following is mentioned: group, table/grid, and collection
4. Explain the concepts of 1D and 2D arrays in programming
5. Have the students decide if each demo item represents a 1D array or a 2D array
6. Take one of the demo examples, display the pseudo-code version of that item
7. Take the empty candy box or a plastic box with the dividers, display the code representation of the box.
8. Using the sticky notes or sheets of paper, write numbers (characters or strings). Display the code to place one of these values into one of the empty locations. Repeat until there are no empty locations

Suggestions:

- The more examples, the better
- Look in cabinets and closets for other 1D and 2D array examples. There are probably many that you normally purchase
- The larger the Legos or Mega Blocks, the easier it is for kids to see from the back of the room (i.e., the size aimed at 1-4 years old)
- With younger children, you may need to lead the brainstorming more than with older children. You can always hint about shape or structure to get the kids thinking in the correct direction
- With smaller groups or older students, I tend to let them select the numbers, characters, or strings being placed in the boxes. With large groups, it can be easier to prep it ahead of time

Basic Networking, Message Passing, and Security with Party Hats and Candy

This activity, described in Figure 9, is designed to teach students basic networking concepts and terminology: message passing, network, node, topology, packet, white hats, black hats, gray hats,

hacking, and network security. All students will have the opportunity to play a node within the network and can be one of three things: white hat, black hat, or gray hat. The activity can be done with almost any size group and generally works better with 5th graders or older. This activity was inspired by an activity in CS Unplugged [2].

Figure 9:
Basic Networking, Message Passing, and Security with Party Hats and Candy

Materials:

- 5-15 White Hats, 1-5 Black Hats, and 1-5 Gray Hats
- Container that can be locked (or can have a chain placed around it)
- Locks (minimally 2) (and chain if needed)
- Small wrapped candy (Starburst, Mini-candy bars, Small boxes of Nerds)

Activity Instructions:

1. Start with 5 student volunteers. Provide each with a white hat. Line the students up in the front of the room in a "straight-line". Explain that as a group they are a network and each person is a node in the network
2. Ask the students if they have ever sent an email. (Most students have and are familiar with the concept). Briefly discuss message passing
3. Using an unlocked container, place a piece of candy in the box. Have the "network" pass the message to the last "node". Re-use the unlocked container (and a new piece of candy) to pass a message back to the sending "node" (in acknowledgment of message received)
4. Add a "node" to the network. Provide the student with a black hat and quietly give them directions that they are an evil node in the network and should "steal" the candy when the container is passed to them (and then continue to pass the container along). Place the student in the middle of the network
5. Re-do step #3, but the candy will be "stolen" twice. Have the students discuss and suggest possible ways to prevent the candy from being taken
6. Re-do step #3 but lock the container. Keep the key (hilarity will ensue). When the candy arrives on the far side, they cannot get into the box. Have the students discuss and suggest possible ways to solve the problem (i.e., private key encryption style). Try various student ideas for message passing
7. Add a gray hat to the network. Discuss the differences between the black and gray hats. Repeat message passing as you add people or change the network
8. Add additional white hats. Repeat message passing
9. Give everyone a hat (randomly) and have everyone stand in a giant circle. Try to arrange "nodes" so that there are long stretches of white hats. Pass messages. Discuss/have students observe how much longer it takes with the larger network
10. (Optional) Discuss possible ways to re-arrange the network to avoid black or gray hats. Have the students re-arrange the network topology. Repeat message passing

Suggestions:

- If you do not want to use candy, then try stickers, smaller erasers, or other small party favor.
- The hats can be of any type (paper, plastic, felt, etc.) and they do not all have to be the same (as computer networks have all kinds of nodes on them)
- You should plan for as many hats as you want students to participate (and generally I have everyone participate) and (minimally) 2/3 of the hats should be white
- You can use any 3 colors of hats. If you do not choose to use white, black, and gray, then it would be best to label the hats with signs
- Gray hats can be challenging to find – an alternative would be a hat with a mix of white and black (e.g., black and white striped, black and white checkers, white hat with a thick black band, or other combinations)
- When working with high school students and older, you can talk about how everything included in a packet (addresses, message, etc.), IP addresses and label each hat with an IP address, and spend more time talking about topologies
- If you have someone assist with the activity, having them initially play the evil role adds to the "surprise"

Learning Objectives: After this activity, students should be able to:

- Define, in their own words, a network (all ages)
- Define, in their own words, a network packet (all ages)
- Define, in their own words, a network topology (high school and up)
- Explain the difference between white hats, black hats, and gray hats (all ages)
- Explain the importance/need of network security (all ages)
- Explain the importance of network topology (high school and up)

Monster's Hate Chocolate: Learning Try/Catch Blocks

This activity can be used to demonstrate the programming concept of try/catch blocks. Additionally, the students practice effective teamwork and problem solving skills. This activity, like the others presented, can be done with a wide age range and with several different programming languages (e.g., Java, C#, C++, JavaScript, PHP, Python, Visual Basic .NET). Figure 10 provides the directions for this activity and Figure 11 is the corresponding instruction sheet.

Figure 10:

Monster's Hate Chocolate: Learning Try/Catch Blocks

Materials:

- Paper lunch bags (two for each group)
- Starburst or other small candy
- Fun size candy bars (or other candy that is larger than your small candy)
- Scissors
- Dark colored marker
- Paper and Printer/Copier to prep the instruction sheet

Prep:

- Draw a monster face on each paper lunch bag
- Cut a month hole the size of your small candy (make sure the big candy does not fit through the hole)
- In a plain paper lunch bag, place the assortment of small and large candies (see worksheet)
- Print/copy the instruction sheet (at least one per group)

Activity Instructions:

1. Have students get into groups of 3-5
2. Have one student from each group come to get the 2 paper lunch bags and the instruction sheet
3. Have the students complete the activity as instructed. When they complete the activity, they should show you the monster bag with only the Starbursts (or smaller candies)

After the students have finished, they can split up the candy

Figure 11: Instruction Sheet

Monster's Hate Chocolate Activity

Instructions: Feed your monster! But the monster is horribly allergic to bad code. Randomly draw a candy from the bag. Use the following code to figure out if you can feed the monster the candy. When you cannot feed the monster, examine the provided code to determine why the monster doesn't like it.

[Red Starburst]

```
int x = 12;
int y  = 4;
int result = x + y;
```

[Pink Starburst]

```
String temp = "12345";
int value = Integer.parseInt(temp);
```

[Yellow Starburst]

```
double[] listOfValues = { 42.1, 9.5, 8.4, 6.83, 36.2 };
System.out.println(listOfValues[3]);
```

[Orange Starburst]

```
int i = 6;
int j  = 2;
int result = i/j;
```

[Reese Tree]

```
String[] groceryList = { "apples", "grape juice", "chips", "salsa", "lettuce" };
System.out.println(groceryList[6]);
```

[Snickers]

```
int num1 = 42;
int num2 = 0;
int resultDiv = num1/num2;
```

[M&Ms or Skittles]

```
String temp = "hello";
int value = Integer.parseInt(temp);
```

[Trix or Butterfingers]

```
int[] myArray = new int[-5];
```

Learning Objectives: After this activity, students should be able to:

- Define, in their own words, a try/catch block (all ages)
- Define, in their own words, an exception (all ages)
- Explain how try/catch blocks work with multiple exceptions (middle school and up)
- Explain the differences between various exceptions (high school and up)

Concluding Remarks and Future Work

The above activities have been successfully used during an introductory computer programming course to teach the concepts of variables, branching statements, loops, arrays, class, and try/catch blocks. The binary/octal/hexadecimal activity is a popular activity and has been used multiple times for classroom visits and outreach tables at a variety of events. Many of the other activities have been successfully used during classroom visits and on-campus school visits to demonstrate networking and message passing. In addition to the computer science knowledge, students have gained valuable teamwork skills. Most importantly, students have expressed enjoyment of these activities. College students have particularly commented on how much these activities have assisted in their understanding and ability to remember the concepts. This parallels the findings in other publications [3, 4] for elementary and middle school students.

The "Basic Networking, Message Passing, and Security with Party Hats and Candy" is extremely popular with the students. This activity has been done with high school students during a weeklong STEM summer camp and with 5th graders during a classroom visit. The basic activity was successfully completed with the 5th graders and the extended version with high school students. During the summer camp time was not an issue and the high schoolers were able to experiment with several network topologies and practiced teamwork problem solving skills. The "Binary, Octal, and Hexadecimal Initial Keychains" has been popular with students and the university administration – which has garnered invitations to various outreach opportunities. The activity is simple enough for Daisy and Cub Scouts and keeps the attention of middle schoolers and high schoolers during several on-campus school visits. When working with the Daisy and Cub Scouts, only the binary and hexadecimal portions are completed, but with the high schoolers the octal portion is added to the activity. When time is extremely limited or when wanting to do more than one activity with the students, often only the binary portion is completed as a bracelet. The more programming themed activities have successfully been used as a method for introducing concepts to an introductory programming course. The course contained 42 students but the activities were easy to accomplish (without losing control of the classroom or needing extra leadership) and students provided positive feedback on each activity.

As the author continues teaching and participating in outreach events, these activities will be used often and others will be designed to cover even more topics in computer science. Each experience with these activities has allowed the author to continually improve and expand the activities for a wider age and experience range. Additionally, the author plans to extend several of these activities to cover more advance computer science topics. For example, with the hat activity, have multiple types of hats each labeled to demonstrate that there are different types of nodes within a network and discuss the role of each. In the future, the author would like to collect data regarding these activities to assist in improving the student experience and maximize their learning.

Bibliography

[1] T. Bell, et al., “Computer Science Unplugged: School Students Doing Real Computing Without Computers,” Computing and Information Technology Research and Education, New Zealand (CITRENZ), vol. 13, no. 1, pp. 20-29, 2009.
[2] T. Bell, et al., CS Unplugged: Computer Science without a Computer. www.csunplugged.org., 2015.
[3] L. Lambert and H. Guiffre, “Computer Science Outreach in an Elementary School,” Journal of Computing Sciences in Colleges, vol. 24, no. 3, pp. 118-124, 2009.
[4] R. Taub, et al., “The Effect of CS Unplugged on Middle-School Students' Views of CS,” Proceedings of the 14th annual ACM SIGCSE Conference on Innovation and Technology in Computer Science Education, July 06-09, 2009, Paris, France, pp. 99-103.