4-2020

Computer Science without Computers: A Guide to Teaching Concepts of Computer Science without Access to Computers or Other Technological Devices

Anna Nagy 20
DePauw University

Computer Science without Computers

A Guide to Teaching Concepts of Computer Science without Access to Computers or Other Technological Devices

Created by Anna Nagy

DePauw University Honor Scholar Program, Class of 2020
Sponsor: Dr. Chad M. Byers
Committee: Dr. Gloria C. Townsend & Dr. Jamie D. Stockton

I would like to thank the following people for their constant support, encouragement, and guidance during this year-long endeavor: Chad Byers, Gloria Townsend, Jamie Stockton, and Kevin Moore.

A special thanks to Amy Welch for her patience during my constant stream of questions.

The biggest thank you of all goes to Chase: thank you for loving, supporting, and keeping me sane through this process. As always, you have been my rock.

Table of Contents

Forward

As the world becomes more are more dependent on technology, jobs in the technology sector are growing at an exponential rate. Computer science is finding itself integrated into every sector of industry, as more programmers are needed for the machines that are automating previously manual tasks. As the world lurches forward towards this technologically advanced future, students need to be prepared for the changes in the job field that will affect their futures.

I had my first interaction with computer science as a freshman in college. I had heard of it at the end of high school, but didn't know what it was and had no clue how important it was. As I talked to others in my classes, I realized that this was a common reality. Even those who had previous exposure to computer science, had not encountered it before at least high school. As I started to look in to this more, I realized that at least in Indiana, the state standards for computer science were weak at best. As I moved through college, I began to volunteer with groups that made an effort to remedy this problem in the local area. We used public library computers and student's personal iPads that they received for schoolwork to play computer games and learn the basics of coding with block code.

However, these activities made me think about all of the math and science camps I went to as a kid. Why were there not any camps for computer science? Why had my high school offered biology, physics, chemistry, and earth & space science but not computer science? Why was this the case for most of my peers as well? This thought stayed in the back of my mind for 2.5 years, and eventually I had thought about it enough to come up with a way to try to start addressing this issue.

The first obstacle that I noticed was time. When state standards do not prioritize learning about computer science, teachers simply do not have time to add activities into their schedule with all of the other topics they need to cover to prepare their students for the many standardized tests they will take over the course of the year. Time is a problem not so easily solvable. This will take more than I am able to provide to be changed, but hopefully soon the importance of computer science will be recognized in schools, and making time will no longer be an obstacle.

The second problem is limited funding for schools prevents teachers from being able to access the technology to teach computer science to their students. Computer labs are only staffed part time because the school requires the staff member to also

be a librarian or secretary. In some schools, teachers are able to fundraise or apply for grants to get tablets for their students, but there are still other classes in the same school that aren't able to receive the same materials.

Lack of resources IS a problem I can address. While I can't find the funding for every classroom, I have decided that the best solution for now is to equip teachers with the skills and knowledge to teach computer science without computers. Although the concept seems strange, I believe it can be equally effective, because writing code on a computer can sometimes be too abstract for younger kids to understand anyway.

The idea came to me as I went through my college classes, struggling to come up with ways to visualize the concepts I was learning. In the beginning classes, professors were very good about using real world examples, and comparing concepts to objects and processes we encountered on a day to day basis, but as I moved into upper level classes that went away, and I had to do that on my own. It was then that I realized that especially with younger kids, hands-on learning is

significantly more effective. Giving kids memories to help remember concepts and helping them link new concepts to things they already know has proven time and time again to be an effective method for learning.

These lessons are designed to use familiar objects and previous knowledge to model computer science concepts and give kids a way to remember how they work and how they are connected.

Lesson 1- What is Computer Science?

Pre-class Preparations:

Print off the images from the Appendix that correspond with Lesson 1. For the first set of 4 images can just be printed off once, to be passed around or hung up on the board for the wrap-up discussion. For the next 2 pages, you should print off one set for each group or table you plan on having.

You should also assemble one craft box per group or table. These craft boxes can include anything you want, but some examples of materials that would be good to include are: pompoms, buttons, pipe cleaners, stickers, construction paper, markers, glue, scissors, and tape.

Beginning Discussion: What are computers?

Have your class split into groups of three or four (larger groups can also work if you have your classroom arranged in pods or tables) and write down some things about what they think a computer is.

Ask them the following questions to have them brainstorm about:

1) What is a computer?
2) What does a computer do?
3) What can a computer look like?

After giving a few minutes of time for brainstorming, have each group share some of their answers with the class. Talk about the differences between really old computer and computers now. After having a class discussion, if some of the following answers have been missed, take a minute to talk about them:

1) A computer is a machine that can solve problems and store information.
2) A computer can search for information, store information, and use information to solve problems.
3) A computer can be a desktop, a laptop, a phone, or many other things

After talking about computers, ask them to think about people who work with computers. Have them get back together in their groups and talk about computer scientists. If they don't know what a computer scientist is or have never heard of one, that's okay! Tell them to give their best guesses based off of their knowledge of scientists and computers.

1) What does a computer scientist do?
2) Where does a computer scientist work?
3) What does a computer scientist look like?

After giving a few minutes of time for brainstorming, have each group share some of their answers with the class. Then talk about the differences and similarities

between each groups' answers. After having a class discussion, if none of the questions have been answered exactly, take a minute to talk about the following answers:

1) A computer scientist can do a lot of things. They can work with large amounts of data to find patterns, write code for computer programs and apps on your phone, and make websites.
2) Computer scientists can work almost anywhere! Lots of different companies today need computer scientists because so much of our day-to-day life involves the internet and other technologies.
3) There is no one look for a computer scientist. They can be men or women, old or young, and from anywhere in the world! If you like problem solving and working with computers, then you might like to be a computer scientist!

For the last part of your discussion, ask students to think of things that they use that a computer scientist might have helped create. This can be anything from smart phone apps to GPS in their cars to the robot vacuum that they use at home. Have them think specifically about the things they might use when they are at school like their library cards or e-report cards.

In-class Activities:

Just like the discussion, there are two parts to the activities for this sessions. The first activity will be an individual, hands-on activity to get students actively engaged, and the second will be a cooperative group activity to get them thinking about all the different types of people who can be computer scientists.

Activity 1 -What is a computer? (20-25 minutes)

For this activity, distribute the craft boxes to small groups of students. Then give the class 15 minutes to each make a model that they think represents a computer. Tell them to remember what computers should be able to do from what you discussed. When 15 minutes are up, gather the class away from the materials and have everyone share their computers with the class. Encourage them to explain in what ways it represents a real computer.

Activity 2- What is a computer scientist? (20-25 minutes)

Give each group the pages of pictures from the Appendix. Ask them to make up personas for each person. Write down a name, age, family, and hobbies for each person. After giving them 10 minutes to do this, tell them to talk about how each person could be a computer scientist. After giving another 5 minutes for discussion, go around the room and have each group share one or two of their people and how they are a computer scientist.

Wrap up:

Pass around the images from the Appendix or hang them on the board. Have each group talk about why people might not picture the people in the images when they think of computer scientists. Encourage them to think about why that's a problem and what they can do to help others think about all the different kinds of people who can be computer scientists. To wrap everything up, have each student write a list of three things that they learned about computers or computer scientists. Finally, have them share with a partner at least one of the things they learned.

Tips for What Might Go Wrong:

When creating a computer model out of arts and crafts supplies, remind students to stay focused on the task and not get distracted by the materials and free creative time. Make sure they are choosing elements for their computers because of the function it represents and not just for decoration. During the second activity, students might have trouble coming up with jobs for their characters based off of the personas they made for each one. Remind them that people can have a job that doesn't relate to their hobbies or what they like to do. Be ready to address stereotypes about what kind of people can like computers and be computer scientists. Remember that debunking those stereotypes is really the point of this

first lesson, so if students don't actually bring them up, you certainly can for the sake of discussing why they are wrong.

Adaptations:

Students who usually require a little extra help may need someone to help them think through the design of their model computer, and help them put it together. For the second activity, if there are students who get overwhelmed in a group, or who you know will not contribute in a larger group, feel free to break groups into smaller ones or change it to partners instead. The same can go for students who are picking things up faster. Put them together and ask them to think about what we can do to help break people's incorrect thoughts about what computer scientists look like.

State Standards:

Science and Math

- SEPS.5 Using mathematics and computational thinking
- SEPS.8 Obtaining, evaluating, and communicating information

English/Language Arts

- 4.RF.1 Apply foundational reading skills to demonstrate reading fluency and comprehension.
- 4.RV.1 Build and use accurately general academic and content-specific words and phrases.

- 4.RV.2.4 Apply knowledge of word structure elements (e.g., suffixes, prefixes, common Greek and Latin affixes and roots), known words, and word patterns to determine meaning.

Lesson 2- Algorithms

Pre-class Preparations:

For this lesson, you will need to print off the graph paper programming worksheets found in the Appendix. There are three levels: intro, regular, and advanced. The goal is for all students to be able to complete the regular or advanced pages, but the into worksheet is also an option for students who need some modification.

For the peanut butter and jelly sandwich activity, if you don't have any peanut allergies you can get the actual supplies for sandwiches and make them at the end. If peanuts are not okay, you can substitute for other sandwich toppings, or skip this part.

Beginning Discussion:

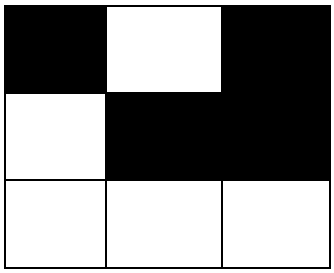
Talk about the new vocabulary word, "algorithm". Write it on the board and have the class repeat it aloud. First ask if anyone knows what the word means. After getting a few guesses, give them the definition "an algorithm is a set of steps to complete a task or solve a problem". Ask the students if they can think of anything else that is a set of steps to complete a task. (Examples are recipes, directions to play a board game, instructions to assemble a piece of furniture, steps for a Lego set, etc)

In-class Activities:

Activity 1- Graph Paper Programming (20-25 minutes)

Start first with a demonstration on the board. Draw a set of 3x3 squares and color a few of them in. Draw a second set of 3x3 squares but leave them blank for now. (These will be used to replicate the first one after you've written out the instructions.) Designate the upper left hand corner as starting square and have the class help you decide what steps you need to take to write out the directions to copy the pattern. You can only move one step at a time in any direction to fill in the correct squares. Write out the directions on the board with arrows indicating what direction you should move, and writing an "X" to indicate that the current square should be colored in. When you have finished writing out the steps, follow them one step at a time to duplicate the first picture in the empty 3x3.

Here is an example of instructions that would help you duplicate the squares below (starting point is the lower left corner): ^, >, X, >, X, ^, X, <, <, X



After doing an example on the board, pass out the Graph Paper Programming Sheets found in the Appendix, and have kids work in pairs to write out directions

for the first few pictures. Then, have them each pick a more challenging picture and write out the directions by themselves. After they have had time to complete the directions, have them trade with their partner, draw a blank set of squares, and see if they can follow the written directions and see if it matches one of the originals.

Activity 2- Peanut Butter Sandwiches (20-25 minutes)

Have students get back in pairs and remind them the importance of remembering all of the little details because with computers, they can't fill in the blanks or make little decisions. Explain how computers have to have every small step written into their instructions because they can only execute directions, not make decisions, or fill in steps on their own. Have them write out the steps to make a peanut butter and jelly sandwich. Tell them that step one is "walk into the kitchen" and then let them go from there.

Collect all of the directions and read them out to the class one at a time. Tell kids to raise their hands when they think the directions missed a step. Go through and "debug" all of the instructions and try to see if any of the directions got it all right. See if there are any steps that everyone missed. For an extra, fun activity, mix up and hand out the instructions to different groups, hand out supplies, and tell each

group to make their sandwich exactly how the instructions tell them to. See which instructions actually make an edible sandwich!

Activity 3- Dance Routine (30-35 minutes)

Start by reviewing the important things about writing algorithms that you have learned so far (breaking things down into small pieces, writing out each step explicitly, and proofreading). Talk about how important it is to write detailed instructions and to check each other's work for mistakes.

Split the class into groups of five or six. Tell each group to pick a dance that they like to do, and write out an algorithm to teach others how to do it. Spend some time going around and helping each group get their instructions written out and having them do the dance exactly following the steps they have written out to see if it works. Then go through the instructions one group at a time with the class. Members of each group go up to the front and demonstrate the instructions while everyone else watches and learn the dances. Then play some music and give the kids some time for a dance party with their new moves.

Wrap up:

Review the concepts of algorithms and debugging and talk about what they are and why they are important. What are the skills needed to do a good job debugging someone's code? Explain that there are some computer scientists

whose job it is to debug other people's code. Compare this to a publisher whose job it is to proofread an author's writing. Talk about why it is so important that code doesn't have any mistakes in it. Why must code be written out with very specific instructions? Computers can't make decisions on their own or fill in the blanks between what is written and what is actually supposed to happen. Ask what was the hardest part for writing an algorithm for something as complex as a dance. What did they do to break it down and make it easier? How did they make sure they weren't missing any steps?

Tips for What Might Go Wrong:

The Graph Paper Programming activity should go pretty smoothly. The trickiest part is keeping track of the steps to make sure that the correct squares are getting filled in. If students are having trouble with this, you can suggest they write the directions for each row next to the row so that they are broken up and not all in one giant line. They can also cross off the steps as they go through them to check their partners work. This can help them keep things straight.

For the Peanut Butter Sandwich directions, it's anticipated that they will miss a lot of the prep of getting out the supplies, opening jars, etc. If most of them do, you can give them a chance to correct their instructions before trading and letting them test the directions by making the sandwich.

Adaptations:

The Graph Paper Programming activity has three levels of worksheets for you to give to your students. There is a regular one, a more advanced one, and one with easier problems for kids who are struggling. This is an easy way to modify the activity, but partners or groups can also be used as well.

For the Peanut Butter Sandwich Instructions, students who are struggling could require more help from you or other adults in the room to think through the steps. Students who are more advanced could try a more complicated recipe, such as baking cookies.

For the Dance Routine, students who are struggling could work on a dance that already has simple, broken down steps like the chicken dance or the Macarena. If students are picking it up quickly, they could write directions for a dance that requires a lot of movement from both hands and feet.

State Standards:

Science and Math

- SEPS.5 Using mathematics and computational thinking
- SEPS.8 Obtaining, evaluating, and communicating information

Physical Education

- Combines locomotor movement patterns and dance steps to create and perform an original dance (4.1.3.A)

- Listens respectfully to corrective feedback from others (such as peers, adults). (4.4.2.A)

English/Language Arts

- 4.RF.1 Apply foundational reading skills to demonstrate reading fluency and comprehension.
- 4.RV.1 Build and use accurately general academic and content-specific words and phrases.
- 4.RV.2.4 Apply knowledge of word structure elements (e.g., suffixes, prefixes, common Greek and Latin affixes and roots), known words, and word patterns to determine meaning.
- 4.W.3.2 Write informative compositions on a variety of topics that provide facts, specific details, and examples from various sources and texts to support ideas and extend explanations.
- 4.W.6.2c Spelling – Using spelling patterns and generalizations (e.g., word families, position-based spellings, syllable patterns, ending rules, meaningful word parts, homophones/homographs) in writing single and multisyllable words.

Lesson 3- Intro to Loops

Pre-class Preparations:

For the Graph Paper Programming, the three levels of worksheets can be found in the Appendix. Print these off before the lesson so that you will have them ready. If you have the space, set up the obstacle course. It would be best to have in a gym or outside, but you could get away with an area in your classroom if you set it up carefully around your space. You will need a series of 3 or 4 objects for students to jump over, something to make a tunnel, a large board and some bricks, and some cones. Using masking tape, make a course on the ground divided into squares like a hopscotch path. Then place the jumps in every other square and the tunnel over 3 or 4 squares. Make a section that is three rows wide and place the cones along the middle row, then have students weave between them. For examples of how the course can be set up, look in the Appendix.

Beginning Discussion:

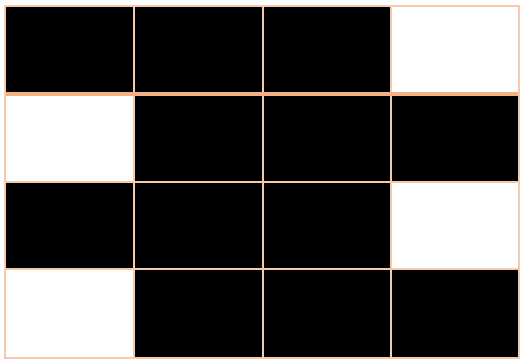
Start off with the generic question of "What is a loop?" without any context. Then ask what a loop might be when related to computer science and coding. Explain that a loop in computer science is a way of writing something that is supposed to happen over and over again (or at least twice). Then ask why they think computer

scientists use loops in their code. Some of the reason are: it makes it shorter, it makes it faster to read/write, and it makes it possible for the number of times to change each time without having to write new code. Ask them if they can think of something they've learned in math that is like a loop. (Multiplication- 3X4 instead of 3+3+3+3)

In-class Activities:

Graph Paper Programming (20-25 minutes)

Here we bring back the graph paper programming activities, but focus on using loops to shorten the amount of instructions you need to write out for each problem. Demonstrate this by drawing two squares on the board again, but this time make them 4X4. Fill in one as follows and write out the directions:



Here, if the starting point is the lower left hand corner, you could write the instructions like this: (>X)3, ^, (<X)3, ^, (>X)3, ^, (<X)3

Talk about how much faster that is to write than writing out each step for every square. Then follow your directions like before and replicate the squares onto the

empty one. After the demonstration, hand out the Graph Paper Programming Sheets found in the Appendix, and have kids work in pairs to write out directions for the first few pictures. Then, have them each pick a more challenging picture and write out the directions by themselves. After they have had time to complete the directions, have them trade with their partner, draw a blank set of squares, and see if they can follow the written directions and see if it matches one of the originals.

Obstacle Course (30-45 minutes)

For this obstacle course, let each student walk through the obstacle course first. Then have each of them will write out directions for how to get through the course, without using loops. After they have written out their initial instructions, have them look at the course and their directions and figure out how to shorten their directions by adding loops. Then encourage them to pair up and go through the obstacle course according to their partner's directions. Tell them to pretend like they're a computer who must follow the directions EXACTLY. That means if there is a mistake in the directions, they still have to do what it says. See how many actually get though the obstacle course without doing something incorrectly.

Wrap up:

Review the reasons why computer programmers use loops: it makes it shorter, it makes it faster to read/write, and it makes it possible for the number of times to change each time without having to write new code. Then ask what challenges students ran into while writing the loops. Talk about mistakes that were made in the obstacle course. What happened when one part of the code was wrong? Did that make the rest of the course directions wrong too? Explain that in this activity, they were just like computers who couldn't make decisions or recognize that what they were told to do was wrong. Remind students about the importance of proofreading their code, especially as they add more complicated elements, like loops.

Tips for What Might Go Wrong:

Again, the Graph Paper Programming activity should go pretty smoothly. Keeping track of the steps to make sure that the correct squares are getting filled in becomes even more tricky when you add in loops. If students are having trouble with this, you can suggest they write the directions for each row next to the row so that they are broken up and not all in one giant line. They can also cross off the steps as they go through them to check their partners work. This can help them keep things straight.

The obstacle course needs lots of space. If possible, outside would be best, but if that isn't an option, a gym, the hallway, or possibly the classroom if you have a large space, will work. Students might have trouble recognizing what to write for the loops in this format, so you may need to go through an example to start.

Adaptations:

Just like the last Graph Paper Programming, there are three levels for worksheets based on the students' skill level. The regular worksheet has a range of difficulties in the problems, but the advanced worksheet has mostly harder problems, and the beginner worksheet is good for those who are learning more slowly. For extra modification, you can allow students who are struggling or learning slowly to work together on the worksheets.

Partner work is also the easiest way to modify the obstacle course directions. You could also modify the activity by working with students who are struggling to get the initial directions for completing the course, and then letting them condense it into loops on their own.

State Standards:

Science and Math

- SEPS.8 Obtaining, evaluating, and communicating information
- SEPS.5 Using mathematics and computational thinking

Physical Education

- Jumps and lands in the horizontal and vertical planes using a mature pattern within activities (such as in dance, educational gymnastics and small-sided practice tasks and game environments). (4.1.2.A)
- Combines movement concepts with skills in small-sided practice tasks (such as gymnastics and dance environments). (4.2.2.A)
- Exhibits responsible behavior in both independent and group situations. (4.4.1.A)
- Listens respectfully to corrective feedback from others (such as peers, adults). (4.4.2.A)

English/Language Arts

- 4.RF.1 Apply foundational reading skills to demonstrate reading fluency and comprehension.
- 4.RV.1 Build and use accurately general academic and content-specific words and phrases.
- 4.RV.2.4 Apply knowledge of word structure elements (e.g., suffixes, prefixes, common Greek and Latin affixes and roots), known words, and word patterns to determine meaning.
- 4.W.3.2 Write informative compositions on a variety of topics that provide facts, specific details, and examples from various sources and texts to support ideas and extend explanations.
- 4.W.6.2c Spelling – Using spelling patterns and generalizations (e.g., word families, position-based spellings, syllable patterns, ending rules,

meaningful word parts, homophones/homographs) in writing single and multisyllable words.

- 4.W.6.2c Spelling – Using spelling patterns and generalizations (e.g., word families, position-based spellings, syllable patterns, ending rules, meaningful word parts, homophones/homographs) in writing single and multisyllable words.

Lesson 4- For Loops

Pre-class Preparations:

Before class, print off the two pages of activity cards from the Appendix and cut them out. You will need to have one copy for each group, so decide how many groups you will want to split your class into, make that many copies, and keep them in separate piles. If you want to make the activity last longer, you can print out two sets of cards for each group. Then gather enough dice for each of the groups to have two six-sided or one twelve-sided die. Two of the cards have them spell a spelling word, as a way to incorporate some practice into the activity, so printing off a list of spelling words for each group so that the kids can pick one would also be a good thing to do for each group.

Beginning Discussion:

Review the concepts of loops in computer science and remind the class why they are used: it makes it shorter, it makes it faster to read/write, and it makes it possible for the number of times to change each time without having to write new code. Then ask them how they think programmers let the computer know how many times they want to loop to run. Explain that there are a few different ways, but one of them is by telling it a set number. This is called a "for loop" because you are telling it how many times you want it to run for. This means if the programmer

wants it to run a different number of times later, they have to go back in and change the number of times that is written in the code.

In-class Activities:

Get Up and Move! (15-20 minutes)

Before leaving the classroom, show the cards and dice to the students. Pull out and example card and write on the board "Do ___ (jumping jacks)" or "Do a (dance party) for ____ seconds". Then roll the dice and put the number in the blank. This is the format of a for loop. Instructions inside of the parenthesis that will be done a set number of times based on what is written outside of the instructions. It is also a good idea to take a few minutes before leaving the classroom to read each of the cards to the class and have students demonstrate each activity so that everyone knows what all of the cards mean. This is also a good time to give modification activities for kids who are unable to do some of the more challenging activities. This activity is best played somewhere outside the classroom where there is lots of space for the students to spread out with their teams and be able to do the physical activities for each turn. If you have access to a gym or to the playground outside, that is ideal, but the hallway or other large indoor spaces will work as well. Divide the students into groups and give each group their dice and activity cards. They will start with the cards upside down in a pile, take turns picking a card and then

roll to see how many time they should complete the activity. The cards represent the directions inside of the loop, and the number they roll is the number of times they are to complete the loop.

Run the activity for five minutes, then mix up the groups, and run the activity again. Do this at least three times with the students in different groups. By doing this you create an even bigger loop. See if anyone notices, and if they don't, be sure to bring it up in the wrap up discussion!

Wrap up:

First ask the class if they realized the bigger loop you were doing while the activity was happening. Then ask them to think of any other loops or patterns in their life that are like for loops; they happen a fixed number of times every time. Some of these can be a week, month, or year that repeats on the same loop every time. In middle school and high school, most students go to several different classes and have different teachers each day, but they have a set number of periods. That daily routine is like a for loop. Sports drills that they run in practice can be a for loop if the coach tells the athletes how many times they are going to run the drill.

Tips for What Might Go Wrong:

When playing any game that involves running or moving around a lot, it is important to be careful of the environment you set up with. If playing outside, be

sure there are no concrete surfaces, holes, rocks, or dips that students could fall or trip on. Proper footwear is also key in preventing injuries in a game with running. Since there are a lot of activities that require jumping and laying down, dresses and skirts are also not ideal.

The biggest thing to be cautious of in this activity is that the kids are taking turns and everyone is getting a chance to roll the dice and pick a card. Hopefully mixing up the groups a few times throughout will help keep this from being a problem, but it is something to watch out for either way.

Adaptations:

The main adaptation for this activity is just physical modification for activity cards that students are unable to do. Give them a few substitution options, or have them do one of the other activities instead. Make sure that groups with students who are struggling are taking their time to complete each physical activity.

State Standards:

Physical Education

- Jumps and lands in the horizontal and vertical planes using a mature pattern within activities (such as in dance, educational gymnastics and small-sided practice tasks and game environments). (4.1.2.A)
- Combines locomotor movement patterns and dance steps to create and perform an original dance (4.1.3.A)

- Combines movement concepts with skills in small-sided practice tasks (such as gymnastics and dance environments). (4.2.2.A)
- Exhibits responsible behavior in both independent and group situations. (4.4.1.A)
- Listens respectfully to corrective feedback from others (such as peers, adults). (4.4.2.A)
- Praises the movement effort of others both more and less skilled. (4.4.3.A)
- Accepts players of all skill levels into the physical activity. (4.4.3.B)
- Exhibits etiquette and adherence to rules in a variety of physical activities. (4.4.4.A)

<u>English/Language Arts</u>

- 4.RV.1 Build and use accurately general academic and content-specific words and phrases.
- 4.RV.2.4 Apply knowledge of word structure elements (e.g., suffixes, prefixes, common Greek and Latin affixes and roots), known words, and word patterns to determine meaning.
- 4.W.6.2c Spelling – Using spelling patterns and generalizations (e.g., word families, position-based spellings, syllable patterns, ending rules, meaningful word parts, homophones/homographs) in writing single and multisyllable words.

Lesson 5- While Loops

Pre-class Preparations:

Before this lesson, gather 10-12 stuffed animals, toys, or other distinguishable items. You should also plan out a large space where you can set these objects out in a long line, about three feet apart. A hallway is a good place for this. Otherwise, you could use the gym or a space outside. The other activity will require a large open space like a gym or outdoor field or basketball court.

Beginning Discussion:

Review the concepts of loops in computer science again, and remind the class why they are used: it makes it shorter, it makes it faster to read/write, and it makes it possible for the number of times to change each time without having to write new code. Then ask them to recall the last type of loop they learned about and how the computer knows how many times to run the loop. After reviewing for loops, introduce the concept of a while loop. Explain that while loops have a condition that tells the computer how long to do the loop. As long as the condition doesn't change, the loop keeps running.

Give the example of being hungry and eating chips. You are going to keep eating the chips, one at a time, in a loop until you are not hungry. Then you stop the loop. Tell the students to get with a partner and think of at least one other thing in their

real lives that is like a while loop. (Examples can include doing their homework one problem at a time until they have finished it, trick-or-treating house by house until their candy basket is full, or searching for items in the grocery store one at a time until they have everything on their list.)

In-class Activities:

Stuffed Animal Game (15-20 minutes)

For this game, you will divide the students up into pairs and allow them each to pick a stuffed animal from the stuffed animals laid out at the front of the room. They should pick the animal in their head, and not tell their partner. Then explain to them the loop they will do. In the hallway, set out the animals three feet apart in a line down the hallway. Students will take turns being the computer who runs the loop: walking three steps, picking up an animal, and checking with their partner to see if it's the correct one. If it isn't, they put it back down, walk three more steps, pick up another, and check again. This process mimics the way a computer would run the loop, check to make sure the condition was still the same, and then run it again. After one partner's animal has been found, have the two switch roles and the other partner play the part of the computer while the first has the animal in their head and checks the condition of the loop.

Red Light, Green Light (20-30 minutes)

This is another game that will require a lot of space to run, but not a ton of prep ahead of time. Before leaving the classroom, explain the rules of the game and how the game simulates a while loop. There will be one student at a time who is "it". They will stand on the far end of the space with their back turned to the rest of the class. They will be in charge of changing the red light to green and back. The rest of the students stand in a line along the opposite edge of the space. They will walk/run forward towards the person who is "it" when that person's back is turned and the light is green, then when the person turns and yells "red light" they have to freeze. The loop they are completing is one step at a time, each time they have to check to make sure that the person who's "it" hasn't turned around or they will get caught, like a computer crashing!

After talking through the rules of the game and the way it relates to while loops, take the class to the place you have selected for the activity and play it for 20 or 30 minutes, rotating the person who's "it" every five minutes.

Wrap up:

Start off the wrap up discussion by posing the question "What is the difference between a for loop and a while loop?" Follow this up with "What about them is similar?" Then talk about why a programmer might want one instead of the other.

(If they know how many times they want something to happen they would use a for loop, but if the number of times they want the loop to run might potentially change depending on the situation, then they would want a while loop.)

Tips for What Might Go Wrong:

Again for this lesson, when playing any game that involves running or moving around a lot, it is important to be careful of the environment you set up with. If playing outside, be sure there are no holes, rocks, or dips that students could fall or trip on. If you do decide to use a basketball court, be aware that students might get more injury from falling. Proper footwear is also key in preventing injuries in a game with running.

Make sure that in the stuffed animals game, students are actually taking three steps between each animal, and actually picking them up each time to keep the loop consistent. It should be more about the process than about getting to the correct animal as quickly as possible.

Adaptations:

For the stuffed animal activity, you may need to be mindful of who is partnered with students who are struggling to understand the concepts. Make sure that those students are paired with someone who will not breeze through the activity, and will take the time to go through it at a slower pace. If you have any students who are

not physically able to run, you can require everyone to walk instead during the red light, green light game. This can actually help keep the game at a slower pace so that students can think about checking to make sure that the person is still turned around.

State Standards:

Science and Math

- SEPS.5 Using mathematics and computational thinking
- SEPS.8 Obtaining, evaluating, and communicating information

Physical Education

- Praises the movement effort of others both more and less skilled. (4.4.3.A)
- Accepts players of all skill levels into the physical activity. (4.4.3.B)
- Exhibits etiquette and adherence to rules in a variety of physical activities. (4.4.4.A)

English/Language Arts

- 4.RV.1 Build and use accurately general academic and content-specific words and phrases.
- 4.RV.2.4 Apply knowledge of word structure elements (e.g., suffixes, prefixes, common Greek and Latin affixes and roots), known words, and word patterns to determine meaning.

Lesson 6- Conditionals

Pre-class Preparations:

For this lesson, you will need to gather plastic clothes hangers for each student, lots of construction paper, scissors, markers, string, and tape for each table or group of students. You will also need straws, sticks, or some other support method for the lower levels of the mobile. You can also gather decorative craft supplies for students to add to their conditional mobiles, but that is optional.

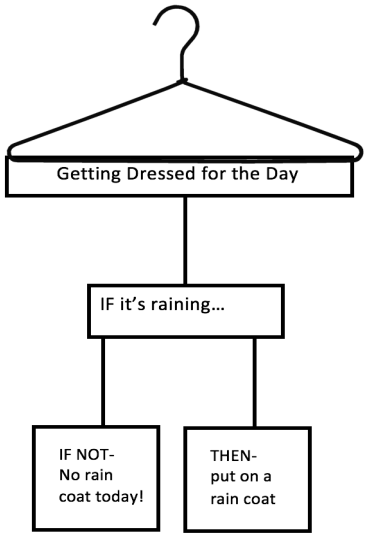
Beginning Discussion:

Think back to learning about while loops. The loop would run over and over while a certain condition was true or false. This created a conditional statement: IF the condition is still true, THEN the loop runs again. These if- then statements help provide options and different paths in the code of a program. Instead of writing a different program for each situation, the code can have a built-in flow chart that tells the computer what to do in both situations. Ask the class if they can think of any decisions that they make that follow the IF-THEN conditional format? (Examples of this could be deciding whether or not they need an umbrella, or whether or not they should put on tennis shoes.)

In-class Activities:

Conditional Mobiles (25-35 minutes)

Put supplies for the activity on each table or split the class up into groups and allow students to work on the floor. Make sure that each student has a hanger, and have them cut out one long rectangle for the bottom of the hanger that describes the flow chart they will be making. This can include getting dressed for the day, picking a pet, or picking a snack. Then, using a piece of string, cut out and attach a rectangle with the first condition to a piece of straw. This could be "IF it's raining", then do the same with two options "THEN put on a rain coat" and "IF NOT no rain coat today". Continue the process with other questions, encouraging them to come up with at least 4 questions that they create options for.



After students have had time to create their conditional mobiles, give them a few minutes to decorate them and make them unique, then have them share either in their groups, or if you have time, with the entire class.

Wrap up:

Take each student's conditional mobile and hang it around the classroom. These can be a reminder of how often we run into conditionals in our lives! To start the wrap up discussion, ask the class if anyone is still confused about how conditionals work. Review how they are related to while loops and why a programmer would use a conditional statement to tell the computer how many times to run the loop. See if anyone has thought of a new conditional statement while working on the activity, and have them share it with the class.

Tips for What Might Go Wrong:

Students might have trouble coming up with ideas for topics to do on their mobile. If that's the case, have the class discuss some ideas together, and tell them it's okay to pick the same topic, as long as they make their conditional statements different. There are lots of decisions to make about getting dressed, so lots of people could make theirs about getting dressed, and then each make decisions about different kinds of clothing. If you do decide to let students decorate their mobiles after,

make sure that they completely finish writing their conditional statements first before they decorate so that they don't lose focus on the point of the lesson.

Adaptations:

This activity can be adapted both for students who are picking up the material quickly, and for those who are struggling. For students who are struggling, you can help them come up with the IF part of the statement, and then let them figure out the THEN and IF NOT parts. If a student is picking up the idea very quickly, see if they can figure out how to do a nested IF statement. For example: IF it's raining, (IF your hands are free, THEN grab an umbrella, IF NOT put on a rain coat) IF NOT no rain equipment today!

State Standards:

Science and Math

- SEPS.1 Posing questions (for science) and defining problems (for engineering)
- SEPS.5 Using mathematics and computational thinking
- SEPS.6 Constructing explanations (for science) and designing solutions (for engineering)
- SEPS.8 Obtaining, evaluating, and communicating information

Physical Education

- Listens respectfully to corrective feedback from others (such as peers, adults). (4.4.2.A)

English/Language Arts

- 4.RF.1 Apply foundational reading skills to demonstrate reading fluency and comprehension.
- 4.RV.1 Build and use accurately general academic and content-specific words and phrases.
- 4.RV.2.4 Apply knowledge of word structure elements (e.g., suffixes, prefixes, common Greek and Latin affixes and roots), known words, and word patterns to determine meaning.
- 4.W.6.2c Spelling – Using spelling patterns and generalizations (e.g., word families, position-based spellings, syllable patterns, ending rules, meaningful word parts, homophones/homographs) in writing single and multisyllable words.

Lesson 7- Types

Pre-class Preparations:

Depending on the in class activities you decide to do with the class, you will need a wide range of materials, and possibly a lot of prep ahead of time. The "Hands-on Type Learning" will require a lot of material gathering. More ideas are listed below in the description for each station, but some good materials to start with are dice, string and letter beads, cards, dominos, light switches, and a number line.

If you decide to do the sorting activity, you will need 10 small, round laundry baskets. Labels for 4 of the laundry baskets can be found in the Appendix. The rest will just be for the teams to start with. You will need 150 plastic ball-pit type balls, made up of 6 different colors. Each team should have their own color balls, but if getting 6 different colors is too difficult, you can draw dots on one side of each of the balls to indicate which team they belong to. Then write a letter, number, word, or "true"/ "false" on each of the balls for each group, spreading between the 4 options as easily as possible. If you want to make the game into a competition, you can get prizes for the winners as well, but that is not the best option for every class, so prizes are certainly optional.

Beginning Discussion:

Explain to the class that when computer scientists write programs, they need different types of answers from their code, depending on the things they want the computer program to do. Compare these different types of answers from the code to different types of answers to questions that you could ask them. Ask the following questions and have each student write down their answers.

1) What letter does your name start with?
2) What is your favorite food?
3) How old are you?
4) Do you like chocolate cake? Then rephrase this question and tell them to write true or false based on the phrase "I like chocolate cake."

Now talk about each "type" and explain how the answer to the question you asked is similar to the answer that computer programmers would want. For the first question you asked, the students should have only written down one letter. This is like a "char", or character type in computer science. For the second question, they answered a word or phrase. That is a "string" type in computer science. The third question should have been answered with a number, which corresponds to a "int" or integer type. The last question was a yes or no, and then a true or false. These are both "boolean" types, which are types that can be one of two answers. As you

discuss each of the types, write them on the board. When you finish talking about each one. Point to them in random order and ask students to raise their hand and tell the class their answer that is the correct type**.**

In-class Activities:

Hands-on Type Learning (20-30 minutes)

This activity is a rotation where you will break the students into four groups and have them rotate around the stations that each have one or more hands on activities to help students understand the different types. You can have each rotation take anywhere from 5 to 10 minutes, depending on how involved your students get and how quickly they seem to be picking up the concepts.

Station 1- Char

Some good materials to have here are scrabble tiles, or pieces from Bananagrams. You can have beads with letters on them here, but they will also be at the "string" station. Another good type of letters to have at this station is plastic refrigerator magnet letters, or foam sticker letters.

Station 2- Int

This station is anything numbers. All different kinds of dice are good. Another idea would be to have a number line, and something that was circular and the right size to encompass one number. Students could use that to highlight one number at a

time on the number line. A sequential counter is another way to physically see integers and by clicking the number up each time they can simulate what might happen to an int in a computer program.

Station 3- String

At this station, beads with letters and some string are needed. Then students can put the beads onto the string to form a string, thinking of as many different words as they can. Have them tie off both ends and make a big list of all of the strings their group is able to come up with. Here you can also incorporate things like practicing their spelling words or writing out sight words.

Station 4- Boolean

This station should only have things with yes/no, on/off, or true/false options. One thing to have is some dominos with one side blank and the other containing one dot. Switches that can be flipped back and forth are also good. Playing cards can be face up where you can see them or face down where you cannot. Coins are a good representation for two options with heads or tails. Any other representations of a two-way choice can go here as well.

<u>Sorting Game (25-40 minutes)</u>

This game is much more high energy, and probably also more fun for the kids. Unfortunately, this is also no a great game to play in the classroom because of the

space required. This would be a great game to play outside, in the gym, or in any other wide, open space that you have access to. Put four of the round, plastic laundry baskets in the middle of the space with labels for each of the four types. Then place the remaining six baskets equally spaced around the outside in a circle. Fill each of the outer baskets with the 25 balls for that team to use to sort. Then divide your students into teams and have each group gather by a basket. On "GO", one student at a time will grab a ball from the team basket, read it, run to the baskets in the center, and decide which basket it belongs in based on what type it is. Then they will run back and tag the next person on their team, who will then pick up another ball and repeat the process. Have each team sit down when they have sorted all of their balls, but once everyone is done, gather everyone in the middle and go through each basket to see if there are any balls in the wrong basket. The goal here is not just to sort the fastest, but also not to put any balls in the wrong basket.

Wrap up:

Because this lesson was all about learning new vocabulary and concepts, wrap it up with a check-in to see if anyone is still confused. Ask the class is there is anything confusing or hard to remember about the different types. Ask what parts of the "Hands-on Learning" helped them remember what each type is. For a final

wrap up, have each student write down 2 things that are examples of each type and share them with a neighbor or their table.

Tips for What Might Go Wrong:

Obviously, when playing any game that involves running or moving around a lot, it is important to be careful of the environment you set up with. If playing outside, be sure there are no concrete surfaces, holes, rocks, or dips that students could fall or trip on. Proper footwear is also key in preventing injuries in a game with running. In games that create competition, it's always best to be careful and watch for bullying or taunting within and between teams. If your class does not handle competition well, you can turn the competition into cooperation but timing each team independently of the others and encouraging the teams to try to beat their first time in a second speed round.

Adaptations:

Specifically, for the Sorting Game, if you have students who are struggling understanding the difference between the types, you can have students go in partners, each grabbing a ball and linking arms to run to the middle. Then have them talk about which type each ball is together, and remind them that they have to agree before putting it in the basket.

There really isn't a need to adapt the Hands-on Learning because each student can explore at the stations at their own pace. If you notice students struggling to understand you could lengthen the amount of time at each station, or again, assign them a buddy to talk through the elements at each station.

State Standards:

Science and Math

- SEPS.5 Using mathematics and computational thinking

Physical Education

- Listens respectfully to corrective feedback from others (such as peers, adults). (4.4.2.A)
- Praises the movement effort of others both more and less skilled. (4.4.3.A)
- Accepts players of all skill levels into the physical activity. (4.4.3.B)
- Exhibits etiquette and adherence to rules in a variety of physical activities. (4.4.4.A)

English/Language Arts

- 4.RV.1 Build and use accurately general academic and content-specific words and phrases.
- 4.RV.2.4 Apply knowledge of word structure elements (e.g., suffixes, prefixes, common Greek and Latin affixes and roots), known words, and word patterns to determine meaning.

Glossary

Algorithm- A set of steps to complete a task or solve a problem

Binary- A way of representing information using only two options (often using the numbers 1 and 0)

Block Coding- Coding that is done by dragging and dropping chunks of pre-written pseudo-code to create a program; a common way to teach beginners how to code

Boolean- A type in programming that indicates an answer that is one of two values (usually true of false)

Bug- An error in a program that makes it run incorrectly or not at all

Char- (Character) A type in programming that indicates an answer that is a single letter

Code- The instructions that programmers create that then tells the computer what to do

Computer- A machine that can solve problems and store information; a machine that can input data, process it, and output new data

Computer Science- The study of computers and computer systems; using the power of computers to solve problems

Conditional Statements- Code that only runs in a certain situation

Data- Information. Usually numbers, letters, symbols, or pixels that are the input or output of a computer program

Debugging- The process of going through code or an algorithm and checking for mistakes, or bugs

For Loop- A loop that runs for a certain, explicit number of times

Int- (Integer) A type in programming that indicates an answer that is a number

Loop- An action or section of code that is executed over and over again multiple times

Program- An algorithm that has been coded into something that can be ran by a computer

Programming Language- A language that can be used to give instructions to a computer that it is able to process

Pseudo-code- Code that is written out in English instead of a programming language; good for beginners or thinking through the process of how to write difficult code

String- A type in programming that indicates an answer that is one or more words

Variable- A placeholder (usually a letter) for a piece of information that can change

While Loop- A loop that runs the entire time a specific condition is true

Additional Resources

CS Unplugged

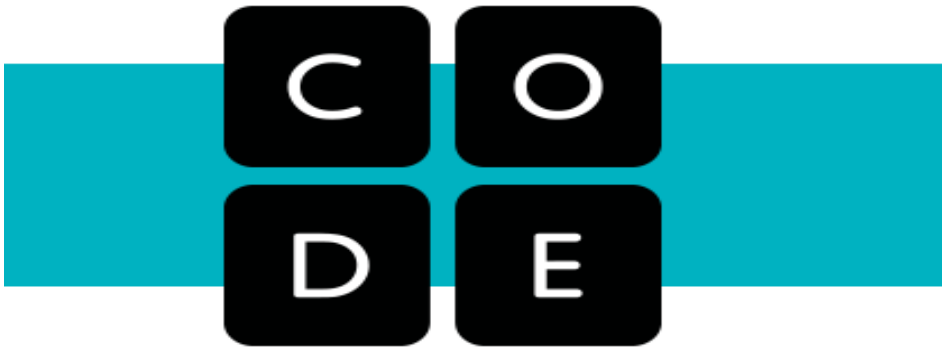
CS Unplugged is an online resource for lessons and activities similar to the ones I have created for this project. The organization was founded by professors at universities in New Zealand and Australia, and has expanded to include members from the United States, Canada, Brazil, Russia, Israel, many countries in Europe, and several other countries all over the world. Their focus is to teach computational thinking, as well as provide a way for computer science topics to be taught without the use of computers and by teachers without a background in the field.

CS Unplugged has lessons for students from ages 5 to 14, including topics such as binary numbers, search algorithms, and sorting networks. Each lesson or activity is connected to other areas of the required curriculum, and there are videos to help teachers understand and prepare for each lesson ahead of time.

Code.org

Code.org is a great resource both for coding practice with computers and learning about computer science without. They are a nonprofit group that focuses on increasing participation in computer science from groups such as women and minorities that are traditionally underrepresented in the field. They provide curriculum for kids K-12, and host extracurricular activities such as Hour of Code, that engages students outside the classroom.

They have tons of lessons on digital citizenship, loops, algorithms, sequencing, and many other computer science topics. Each lesson has a detailed lesson plan, and many others have lesson videos, teacher prep videos, and answer keys. There is also a curriculum guide that can be downloaded that allows teachers to see the lessons in each course laid out, as well as core concepts, attitudinal goals, and teaching tips that can help teachers who don't have previous experience with computer science.



Computer Practice

Hour of Code

Hour of Code is a website that branches off of Code.org, and focuses on letting students learn about programming and computer science outside of school. However, it would also be a great resource for a day in the computer lab where each student has access to their own computer.

After going to hourofcode.com, go to the "Activities" tab on the top right, and you will be taken to a page with hundreds of online coding games. The activities range from pre-reader to grades 9+, and can be sorted by age, activity type, length, and available technology (computers, iPhone/iPad, Android, and poor internet). Games have a variety of themes and characters, so everyone can find something they like. These include themes like animals, musical instruments, and cars, and characters from Star Wars, Barbie, superheroes, and Disney movies.

Scratch

Created by the Lifelong Kindergarten Group at the MIT Media Lab, Scratch is a completely free program that uses drag and drop block code to help kids learn how to program. The MIT Media Lab is a branch of MIT's School of Architecture that has taken on the challenge to conduct research that does not cling to traditional academic topics, and instead incorporates technology and art into their projects. Scratch is one of their most successful projects; currently, it is used in 150 countries, and can be used in 40 different languages.

The platform allows students to both engage in free play and create their own stories, as well as follow along on specific tutorials that give step-by-step directions on how to code games and stories. While Scratch does not focus as heavily on learning the concepts that some of the other sites teach, it is a great resource for starting to learn how to code after a basic understanding of the other principles are understood.

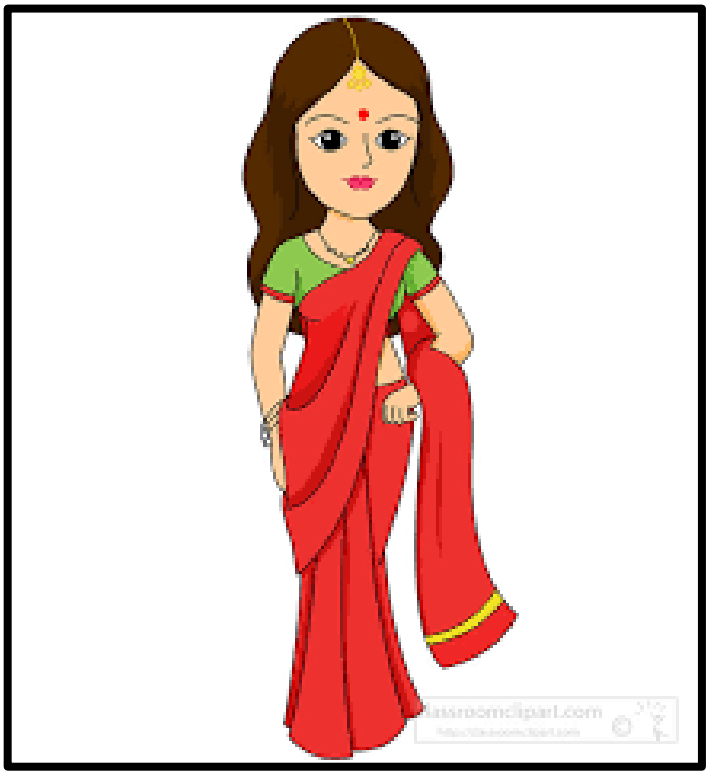
Tynker

Tynker was created by a group of parents, developers, educators, and gamers, who saw the need for a way to introduce kids to coding in a more engaging and exciting way. They have offices in California and Chicago, and have recruited several investors from Silicon Valley. While this site is not as accessible because it only allows you a short free trial before you have to start paying for the content, it still has some pretty cool content.

As a student or teacher, you can sign up for a free account and get access to several different coding games and puzzles, but after a certain number of free games you have to pay to unlock the rest. These games are very similar to the games provided by Hour of Code in that they use popular character and themes to get kids excited about coding. Tynker also uses Minecraft's popularity to get kids trying to learn how to code. They offer skin designing, learning how to make mods, and creating texture packs.

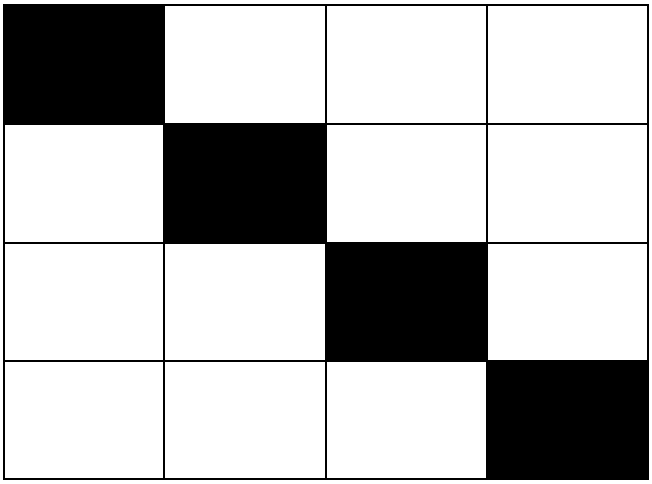
The Appendix

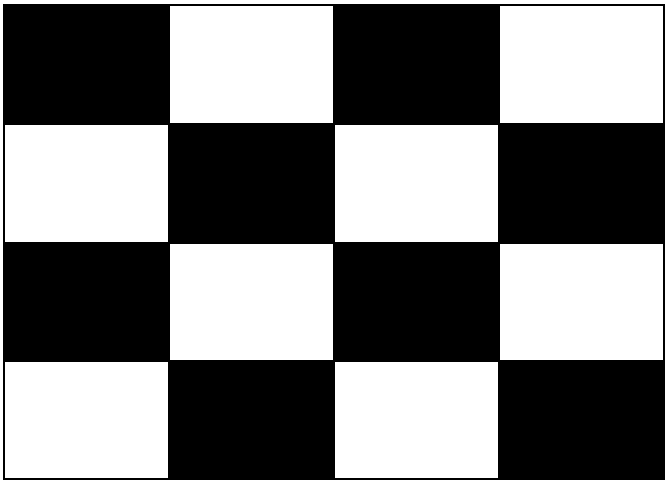
Lesson 1

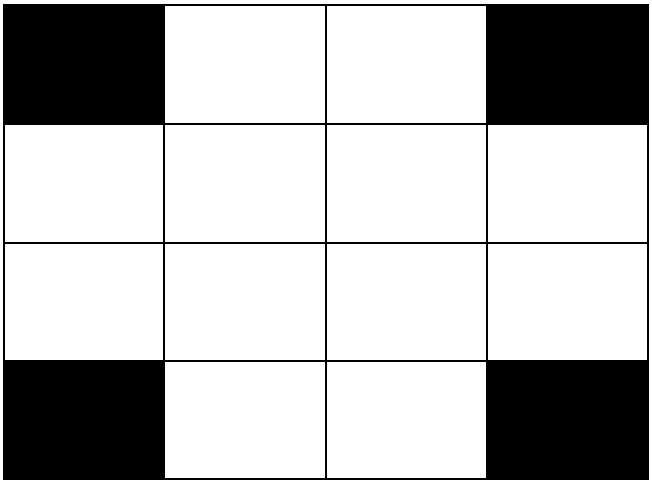


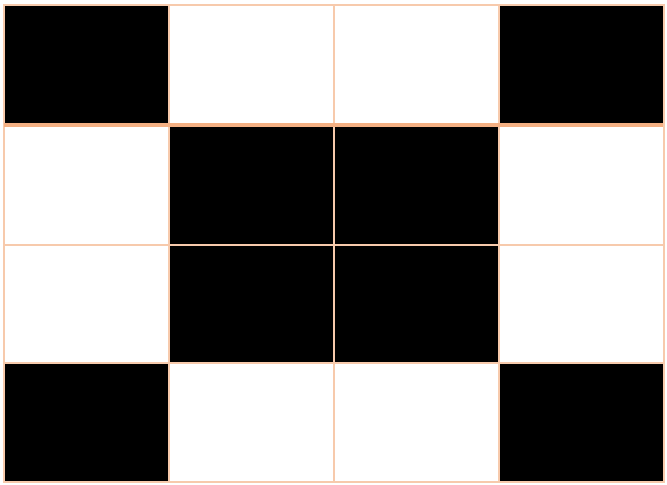
Lesson 1

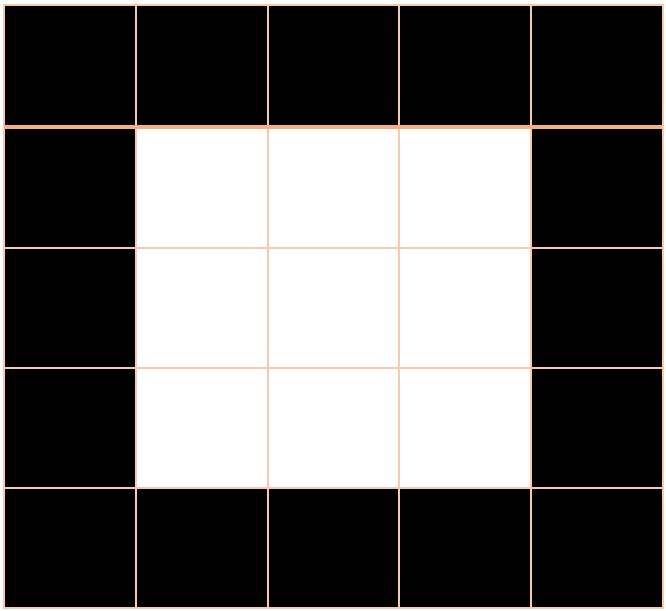
Name: _____

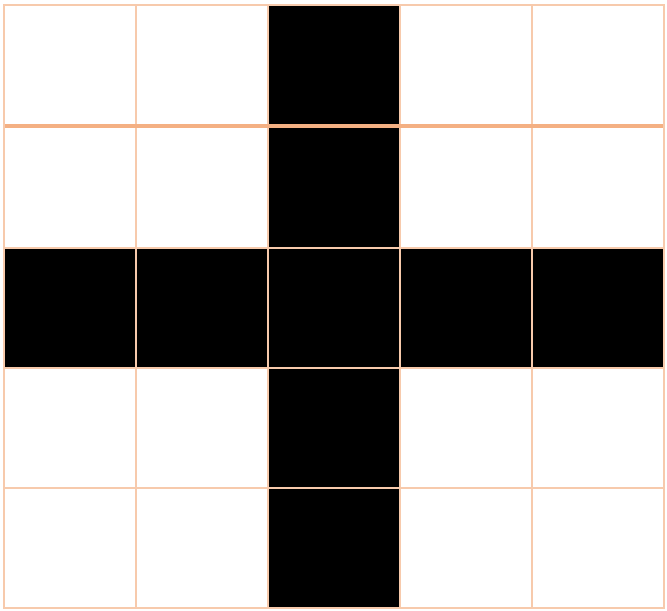




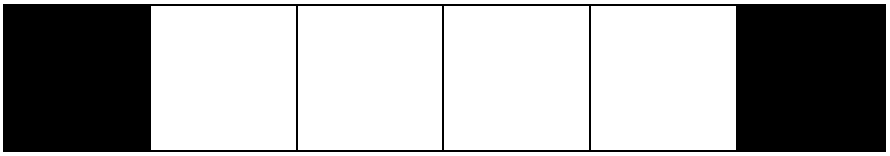


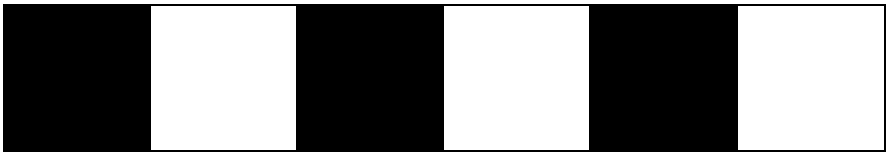


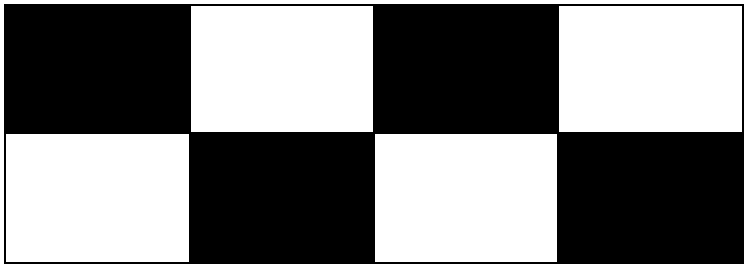


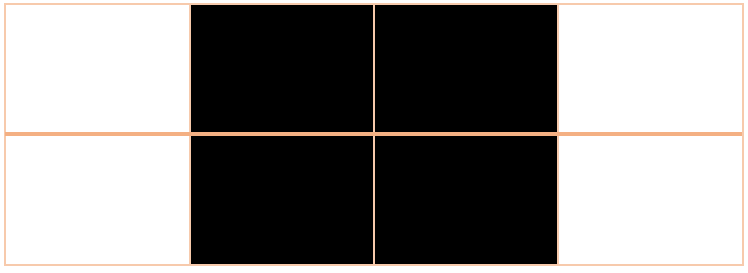


Name: _____

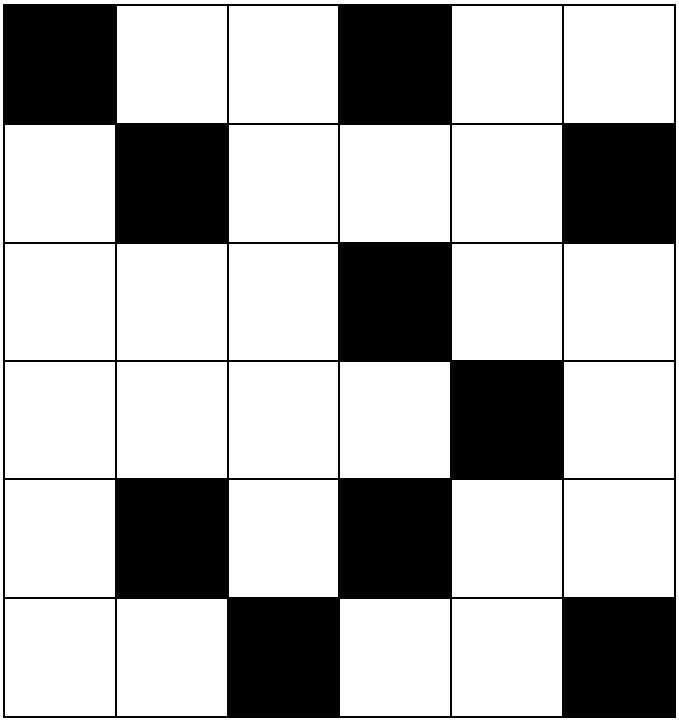


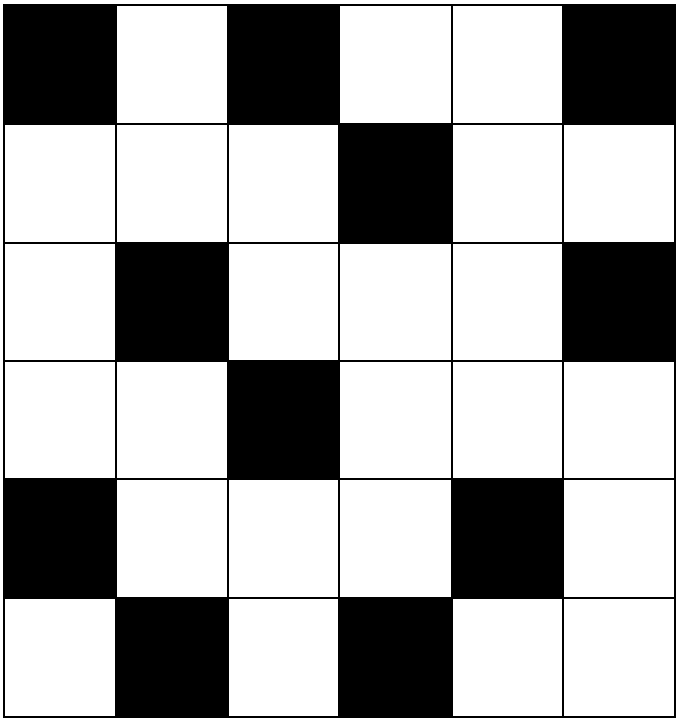


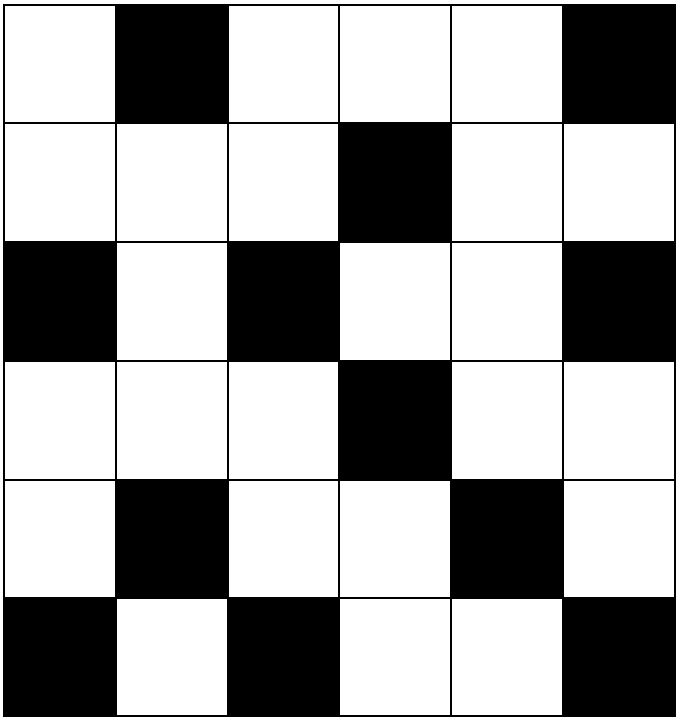


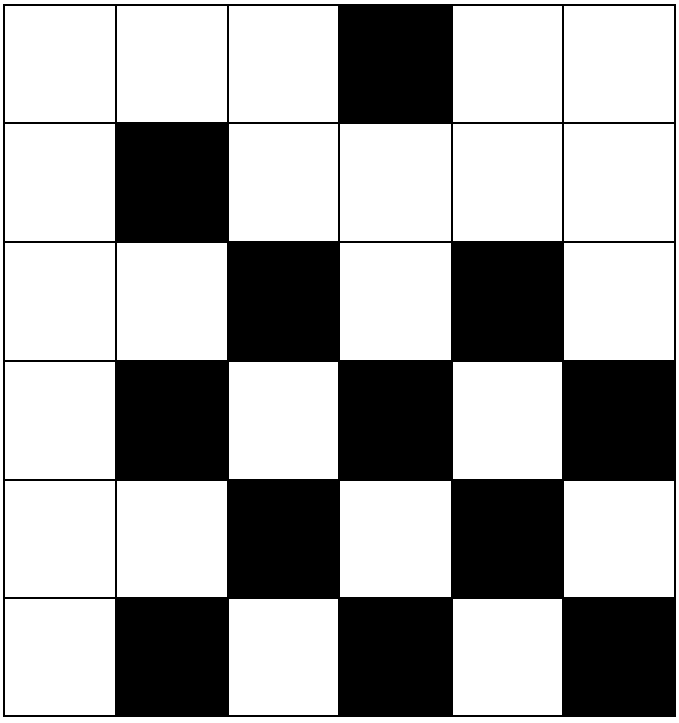


Name: _____

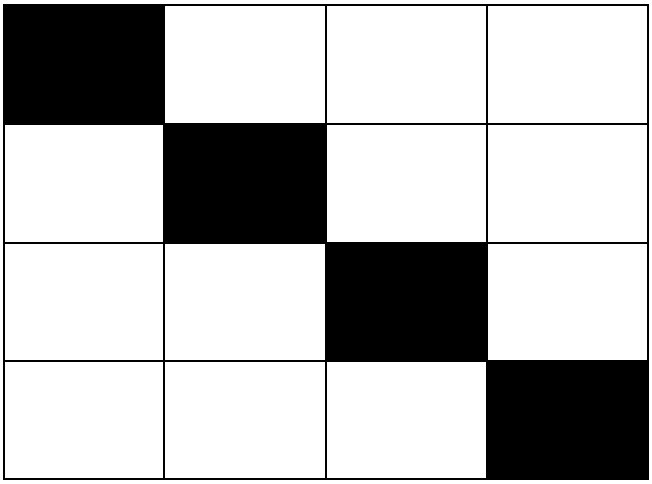


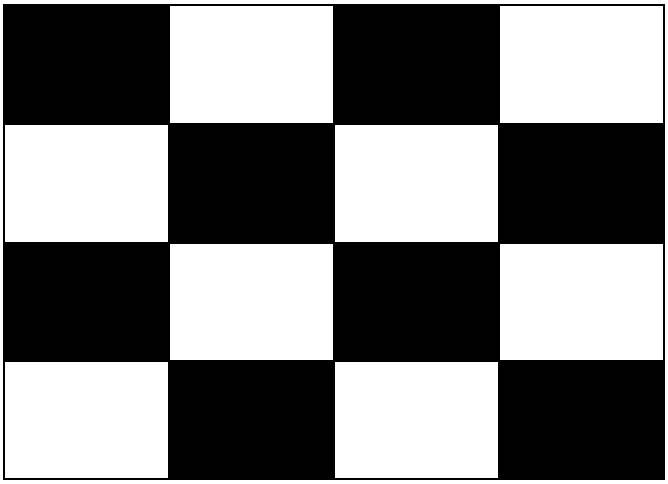


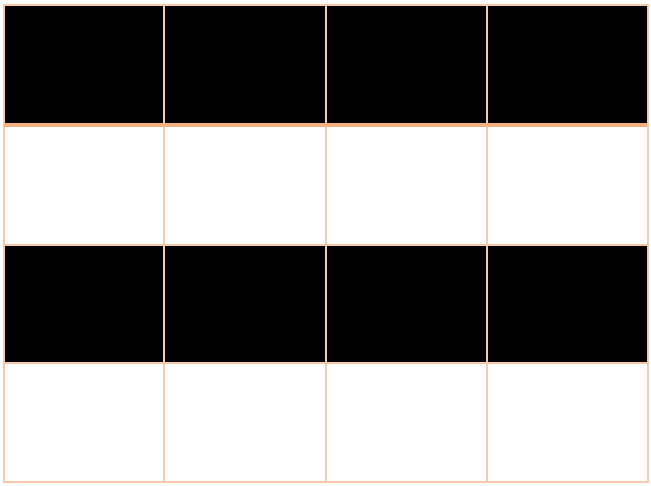


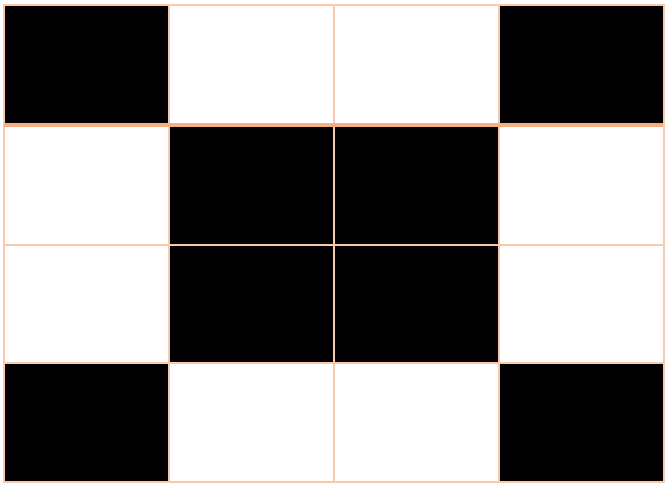


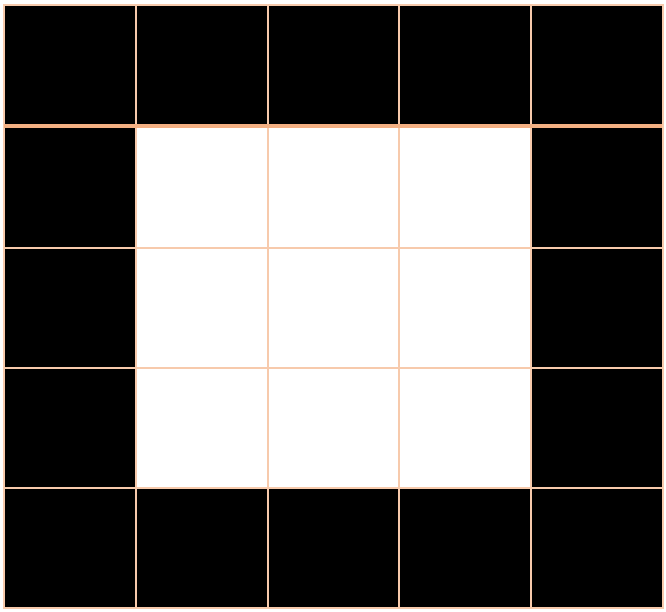
Name: _____

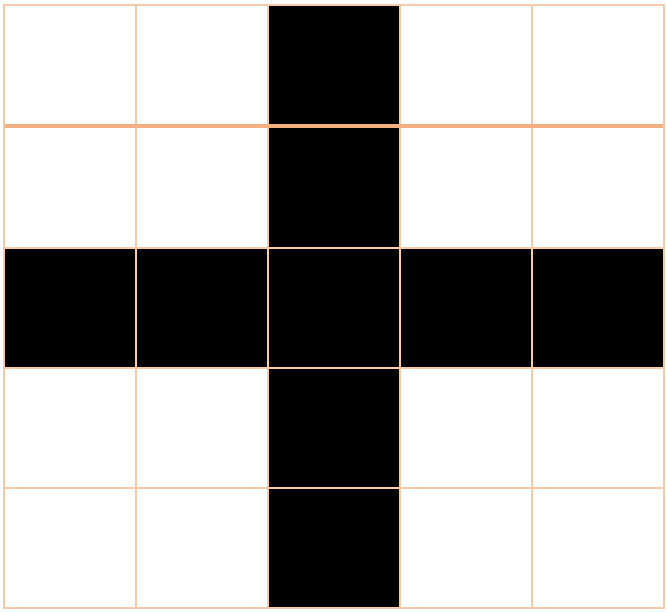




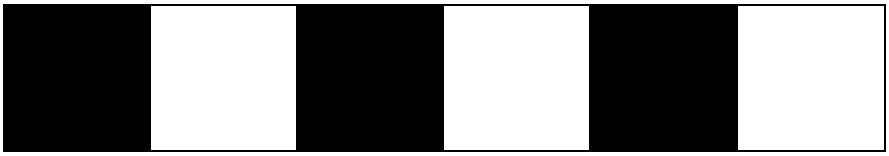


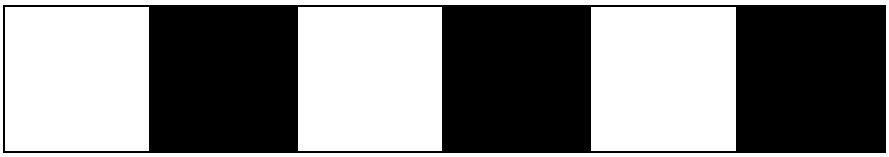


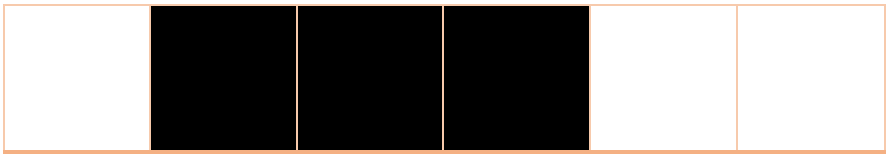


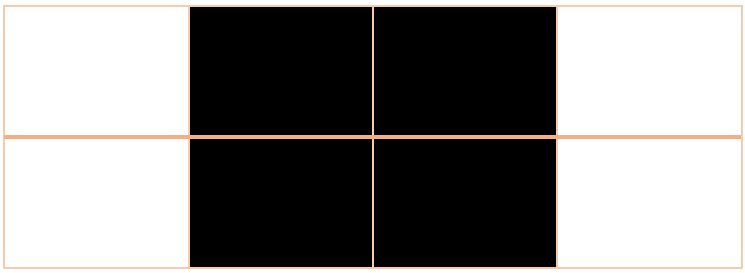


Name: _____

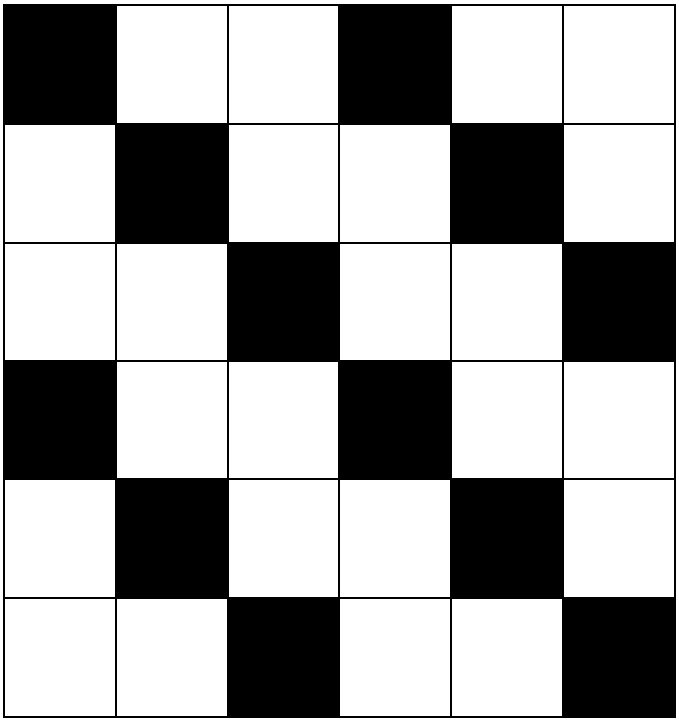


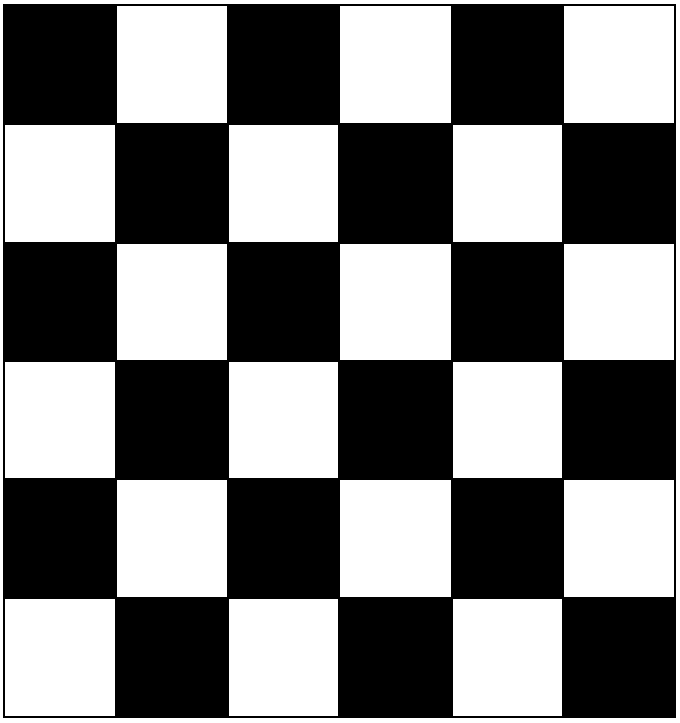


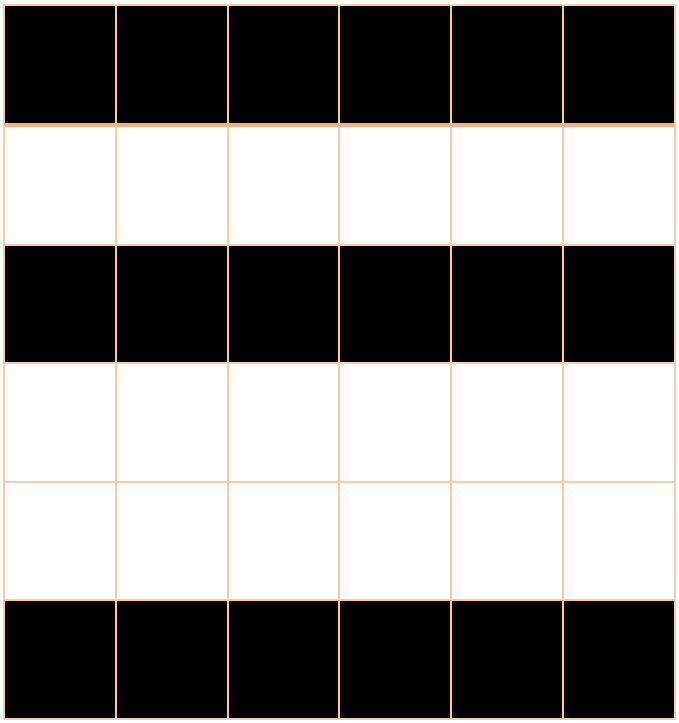


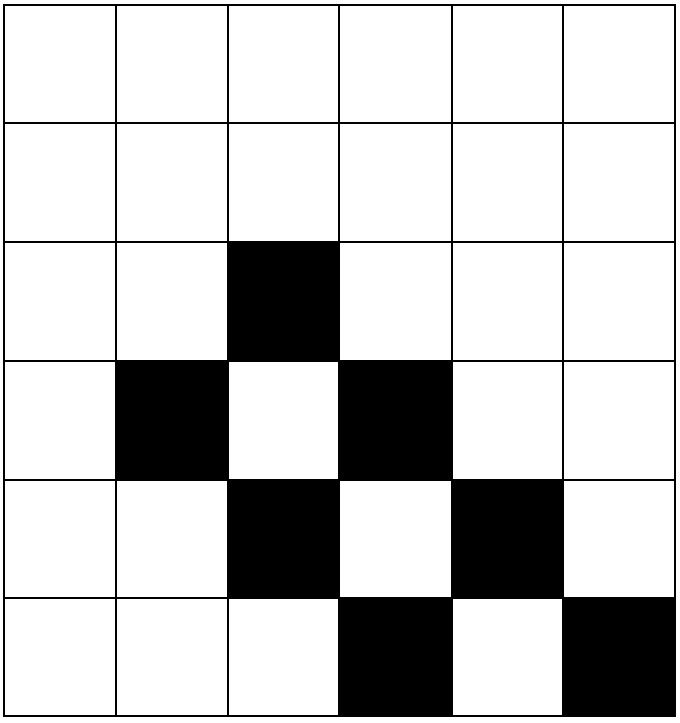


Name: _____



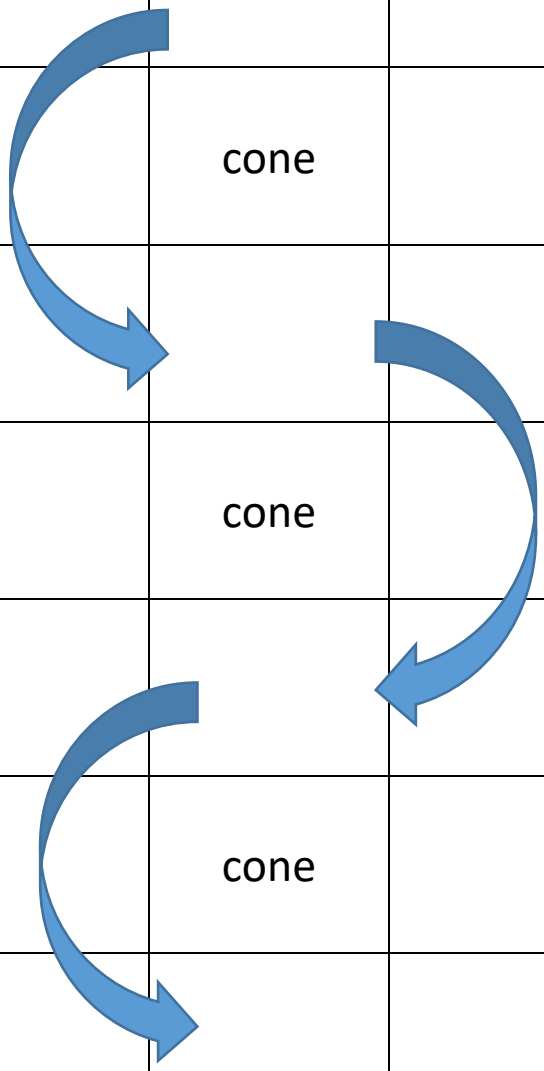






Optional Obstacle Course Map

↓					←
		cone			
jump					↑
		cone			Raised plank
jump					Raised plank
		cone			Raised plank
jump					Raised plank
→	tunnel	tunnel	tunnel		↑



Jumping Jacks	Pushups
Situps	Arm Circles
Dance Party	High Knees

Jog in Place	Toe Touches
High Fives	Spelling Word
Supermans	Spelling Word

Int

(Integer)

Char

(Character)

String

Boolean