How to Complete a Computer Science Project

What is a computer science project?

A computer science project uses coding language to develop information processes or programs to demonstrate, analyze, or control a process/solution. Sometimes robots or intelligent machines are used to use the coding language and perform tasks.

1. Get an Idea for Your Computer Science Project

Like a science fair project, a computer science project starts with a problem, but the problem is a bit different. In science, you might be asking a "What if?" question, such as "What will happen if I add food coloring to saltwater before I evaporate the water?" Computer science, on the other hand, looks at the real world, sees a problem, and uses coding language to try to solve the problem. In other words, what do you see in the real world that you think you can fix, change, or improve? Examples could include developing an application, designing a game, writing a program for a robot, or programming a microcontroller (Raspberry Pi, Arduino, AdaFruit Circuit).

There are two categories in the Computer Science division of the Science and Engineering Fair.

- ***Robotics and Intelligent Machines*** projects use machine intelligence to complete a task or reduce the reliance on human intervention. If you have an interest in computer science, you might look at a career in:
    - biomechanics
    - cognitive systems (artificial intelligence)
    - robot kinematics (how robots move)

- ***Coding*** focuses on the study or development of software, information processes or methodologies to demonstrate, analyze, or control a process or solution. Learning to code could lead to a career in many fields, including:
    - algorithms
    - cybersecurity
    - databases
    - programming languages
    - operating systems
    - machine learning
    - application development

2. Start Programmer's Log Book

A detailed Programmer's Log Book with accurate records allows programmers to describe their coding processes and reflections on program development and debugging so others can follow the process. Your log should be a bound notebook (such as a composition book). It should be

done fully in ink. That's because it can be used as a "legal document" to prove your code is your creation. In the real world, the Programmer's Log Book is used as proof for patents and copyright. It can even be used as evidence in lawsuits over who was the first person to come up with a new idea. That's a pretty powerful book!

Don't worry about making mistakes or making a messy drawing. Mistakes are part of the process of learning and discovering. If you make a mistake, just draw one line through the mistake and keep going. Don't tear out pages or scribble out anything. It's possible that a string of code you thought wouldn't work early in the process turns out to be the solution to your problem.

Setting Up Your Programmer's Log Book

- You will only have one section in your log book so it is very important that you record detailed notes about the work you complete on your project each day. Each entry will have two parts: **Daily Work** and **Daily Reflection**. Be sure to label each part for every entry that you make.
- If you make a mistake, draw a line through it and re-write it. Do not erase or white out a mistake.
- In the **Daily Work** part of each entry, write down all the things you do or plan concerning your project each day. ***Make sure you date every entry.*** Think of it as a blog post each day:
    - What did you do today for your project?
    - Did you record any ideas for your program (sketches of characters, tasks for your robot, story ideas for your game, input/output for your microcontroller)?
    - Did you change any of your code today? Did you take screenshots before and after you made changes to your code?
    - Who did you talk to about your project?
    - What did you research? Make sure to add the bibliography information for each source.
    - Give details! Each day's entry should show the progress on your project.

- In the **Daily Reflection** part of each entry, think about what you learned today:
    - What roadblocks or obstacles did you run into today?
    - What resources did you use to solve your problem (tutorials, asking a teacher for help, looking up code)?
    - If you made changes to your code, what did you learn from it? How will your new learning help you be successful next time?
    - What new ideas or questions have come about as a result of working through the roadblock or obstacle?
    - What successes did you have today?
    - Did your successes spark new ideas for your code/program?

- Why do you think what you learned is important?
- Do you notice any patterns or repeated structures in your code?

3. Complete the Project Approval Form

This form lets your teacher know what you've chosen for your project. It gives an overview of your project with enough detail that anyone who reads it can get a pretty good idea of what you will be doing. Once your teacher approves the project, he/she will give this form back to you. It will have a list of other forms you will need to complete for your project. ***Make sure you keep this signed form and all forms you complete--they are required to be with your project.***

4. Become an Expert in Your Problem

The research phase of your project is very important. This is where you learn everything you can about the topic of your project. If you are trying to solve a problem, you need to understand the problem. Spend some time getting background information. Good research will help you become an expert on your topic. Remember to write down the bibliographic information about each source you read, consulted, or tried to contact. Some ideas of places to go for research are:

- library
- internet--Make sure it is a ***reliable*** source of information (talk to your school media specialist about this).
- experts in the field
- Write to companies involved in your field.

5. Complete Ethics Agreement and Risk Analysis and Designated Supervisor Form

By signing the *Ethics Agreement*, you are saying that you won't copy someone else's work. You can refer to someone else's work, but you have to cite it in your log and on the bibliography. Copy-and-pasting images, words, etc.from the internet is considered plagiarism. If you identify *where* you got each part of what you copied (cite the source), you have done your job.

The *Risk Analysis and Designated Supervisor Form* is used to state all the risks in your project. Risks might include:

- the materials and programs you are using. How can you stay safe when you use them?
- the location you are testing in. Is it close to a road or body of water?
- the tools you may use if building a robot or other intelligent machine.

In this handbook, the Risk Assessment and Safety Considerations section will help you complete this form.

6. State the Problem in a Question Form

Your project problem is how you will develop a program using a coding language to solve a problem. The problem should be a practical need. Are you coding a completely new program or are you modifying (changing) existing code to make it work better in certain conditions? Whatever it is you are trying to do, your final program should be a process/solution to the problem you identified. Your problem should also be very specific. For instance, if you want to design a game, be very specific about which coding language and tasks your program will perform. For example, you might ask, "How can I use Scratch to design a chase style game?" Also, be sure to consider real world applications of your program.

7. Research

Computer scientists need to get a full picture of the problem they are addressing before they start developing their programs.That's where research comes in. For example, if you are programming a robot, find out the coding languages that are compatible with that robot. If you are using a microcontroller to program circuits, research what you will need to build the circuits, how the parts of the microcontroller operate, and the most efficient coding language for the microcontroller. Research helps you to fully understand the problem and possible solutions before you start your design.

For the Science and Engineering Fair, at least ***3 sources*** are required for the research phase. These sources must be documented in the both Programmer's Log and on a bibliography. Interviewing a computer programmer or other expert in the field of your project is an acceptable source.

8. Develop a Project Goal

Your project goal should start as a brainstorm of several solutions/processes to your problem. Don't stop at just one. Brainstorm alternative solutions/processes that might solve the problem, then choose the one that you think best fits the specifics of your project goal. At judging time, you will be asked about different ideas your brainstormed and why you thought your solution/process was the best. All of your solutions/processes should be in your log book, with detailed labels and components of your program. Programmers might include designs of a maze a robot navigates through, sketches of a character they are developing for Scratch, illustrations of circuits, or a display menu for an application .

9. List Materials and Programs

Include any materials you plan to use, including specific robots, devices, and materials you need to complete tasks when appropriate (tape, construction paper, batteries, sensors, wire, LED lights, etc.). Also include a list of the programs and coding language you will use (Scratch, Arduino IDE, MakeCode, Tynker, JavaScript, HTML5, Xcode, C++, etc.).

10. Write an Algorithm (Step-by-Step Procedure)

An **algorithm** is a to-do list for a computer. A recipe is a good example of an **algorithm** because it tells you what you need to do step-by-step. It takes inputs (ingredients) and produces an output (the completed dish).The algorithm is your procedure to developing your program. Using statements, write the steps you will need to code to perform the tasks for your solution/processes. Your algorithm could be written as an outline, a list of steps, in a flow map, or in a storyboard.

11. Develop/Test/Debug/Modify the Program

Using your algorithm (step-by-step procedure), write your code to perform the tasks of your project goal. Good programmers run their programs after they write each line of code. They are testing that the code runs correctly. If an error in the code is discovered, it is easier to find the error in the string of code when you test your program frequently. Finding and correcting errors in your program is called **debugging**.

On your board, you will be required to display changes you have made as you develop your program. Screenshots will help you document these changes. Projects should include screenshots of your initial program, several changes as you debug and modify your code, and your final program. You also might want to take screenshots of strings of code that you feel are significant to your project goal, a complicated design, or were challenging to develop.

12. Final Reflection

Your final reflection should demonstrate your thinking about what you have learned. You could create a timeline with descriptions or steps in the process that show the creation of your project from start to finish. Discuss lessons learned from your project, including ideas you have for future research, steps or processes you would do differently, and other lessons you have learned that will help you with your program next time.

- Tell the story of your project. Why did you choose to create this digital product?
- Describe the design process you used in your project. Imagine you are telling a classmate about it who is interested in working on a similar project.
- Tell the story of how you solved roadblocks or obstacles that came up in the development of your program. Give examples of a few challenges you encountered and how they created problems in your code.
- What resources did you use to problem solve?
- How did the ideas for parts of your program change after you started?
- What new ideas, questions or goals have come about as a result of your work on this project?

13. Communicate Your Results/Construct a Display

Computer scientists share their findings with others. If your program solves a problem, it is good to let others know about it! You should be able to fully explain all parts of your project:

- How did you come up with the problem?
- Explain why you chose this coding language (Scratch, Arduino IDE, MakeCode, Tynker, JavaScript, HTML5, Xcode, C++, etc.).
- Describe the process you used for debugging your program.
- Share a specific string of code, and explain its importance in the program.
- Which string of code are you most proud of, and why?
- What modifications did you make? Why did you make them?

Below is a sample of a Computer Science Project Display Board. Your board does not have to match this exactly, but it MUST have your problem and tell the story of your project.