Name:

Create PT - Written Response Template

A.P. Computer Science Principles

Program Purpose and Development: Prompts 2a, 2b, 2c, and 2d

|  |
| --- |
| Video: Submit one video in .mp4, .wmv, .avi, or .mov format that demonstrates the running of at least one significant feature of your program. Your video must not exceed 1 minute in length and must not exceed 30MB in size  Prompt 2a: Provide a written response or audio narration in your video that:   * identifies the programming language; * identifies the purpose of your program; and * explains what the video illustrates.   *(Must not exceed 150 words)* |
| I used the Processing programming language based on the Java language. The purpose of my program is to draw and animate a robot using the Processing development environment. The video illustrates the code of the program then shows the code running. The video showcases the different functions of the robot responding to user input. The responses to the input include text output, movement, color changes, and special events. The video showcases any and all events to the user, allowing the viewer to get a good idea of all the functions the program has to offer. |

|  |
| --- |
| 2b. Describe the incremental and iterative *development process* of your program, focusing on two distinct points in that process. Describe the difficulties and / or opportunities you encountered and how they were resolved or incorporated. In your description clearly indicate whether the development described was collaborative (i.e. somebody helped you) or independent. At least one of these points must refer to independent program development.  *(Must not exceed 200 words)* |
| This development was individual.  I first started the development process by “drawing” the robot in Processing. The robot is a combination of rectangles, circles, and ellipses that are different colors. I output the shapes to the screen and tweaked the parameters as needed to form the robot. There was some difficulty in scalability and proportions, but trial and error methods were able to fix these issues.  After the robot was output to my liking, I then worked on animating the robot. I started with the setup and draw functions to constantly refresh the robot. I set the frame rate and output window size, then went to work on the animation. I started by animating the side-to-side movement of the robot, then the color change functions of the body and face, and finally the special events. The side-to-side movement are key presses that change the X-coordinates of the robot based on specific keys. The color change events are mouse events that, when clicked, generate a random color, or change the face color momentarily to a preset color. The special events included mouse and key-press events that output words to the screen and changed colors and “missiles” launching from the robot. These events were met with some proportion errors compared to the robot, but by using some of the information from the robot and the previous functions, these issues were able to be solved. |

|  |
| --- |
| 2c. Capture and paste a program code segment that implements an algorithm (marked with an oval in Section 3) and that is fundamental for your program to achieve its intended purpose. This code segment must be an algorithm you developed individually on your own, must include two or more algorithms, and must integrate mathematical and/or logical concepts. Describe how each algorithm within your selected algorithm functions independently, *as well as in combination with others*, to form a new algorithm that helps to achieve the intended purpose of the program.  *(Must not exceed 200 words)* |
| |  | | --- | | Code Segment |   **1.)**  //checks if a key is pressed  if(keyPressed){    //if the 'd' key is pressed then the robot moves to the right  if (key == 'd' || key == 'D'){    XChange += 10;  }    //if the 'a' key is pressed, the robot moves to the left  else if(key == 'a' || key == 'A'){    XChange += -10;  }    //if the 's' key is pressed, the arm moves down  else if(key == 's' || key == 'S'){    armChange = 100;    }    //if the 'w' key is pressed after the 's' key has been pressed, the arms move back up  else if(key == 'w' || key == 'W'){    if (armChange == 100){    armChange += -100;    }    //if missile is true and the 'w' key is pressed, the missiles 'fire' straight up  if(missile){    missileChange += -100;    }    }    //if the 'e' key is pressed, the missiles are created at the robot's side  else if(key == 'e' || key == 'E'){    missile = true;  textSize(32);  text("Missiles activated", 100, 100);    }    //if the 'p' key is pressed, the please boolean is activated to create the text that the user chooses to follow the mouse  else if(key == 'p' || key == 'P'){    please = true;    }    //if the 'q' key is pressed, the boolean please is deactivated  else if(key == 'q' || key == 'Q'){    please = false;    }  }  **2.)**  //checks if the mouse is pressed  if(mousePressed){    //if the left button is pressed,  if(mouseButton == LEFT){    fill(#C6222D);  circle(width - 550 + XChange, 150 + YChange, 25);  circle(width - 450 + XChange, 150 + YChange, 25);    fill(#FFFFFF);  circle(width - 500 + XChange, 200 + YChange, 50);  fill(#08F5FF);  circle(width - 500 + XChange, 200 + YChange, 30);    }    else if (mouseButton == RIGHT){    r = random(255);  g = random(255);  b = random(255);    }  }    if(please){    if(keyPressed){    if(key == 'r' || key == 'R'){    textR = random(255);  textG = random(255);  textB = random(255);    }  }    textSize(32);  fill(textR, textG, textB);  text("we vibing bruh", mouseX, mouseY);    }  }   |  | | --- | | Written Response |   The first algorithm used is a key press algorithm that checks specific keys using logic. The if and else statements first check if a key is pressed. The once true, the succeeding if and else's check the specific key presses for events. This runs through an algorithm by using a step by step process to check keys and then run events based on those key presses. The checks also ignore case of the key presses by checking for capital and lowercase check cases.  The second algorithm used is a mouse button press algorithm that checks mouse buttons. Similar to the key presses, the algorithm uses an if statement to check if the mouse is pressed, then follows up by checking the exact button pressed. These specific button presses also lead to events, similar to the key press algorithm. This is a step by step process to check the buttons on the mouse and lead to the mouse specific animation events on the robot.  Both of these algorithms implement checks on the user input and can be used in conjunction to change the color of the robot while moving it for example. These algorithms use similar structures and check some events from the other to then start their own events. |

|  |
| --- |
| 2d. Capture and paste a program code segment that contains an abstraction you developed individually on your own (marked with a rectangle in section 3). This abstraction must *integrate mathematical and logical concepts*. Explain how your abstraction helped manage the complexity of your program.  *(Must not exceed 200 words)* |
| |  | | --- | | Code Segment |   void draw(){  background(255);    //head ellipse  fill(#254472);  ellipse(width -500 + XChange, 200 + YChange, 250, 100);  //body rectangle  fill(r, g, b);  rect(width - 650 + XChange, 251 + YChange, 300, 400);  //arm / leg rectangles  fill(#8B6C9B);  rect(width - 750 + XChange, 600 + YChange, 100, 200);  rect(width - 350 + XChange, 100 + armChange, 100, 200);  rect(width - 750 + XChange, 100 + armChange, 100, 200);  rect(width - 350 + XChange, 600 + YChange, 100, 200);    //antennae circles  fill(#72AB98);  circle(width - 550 + XChange, 150 + YChange, 25);  circle(width - 450 + XChange, 150 + YChange, 25);    //eye circle  fill(0, 0, 0);  circle(width - 500 + XChange, 200 + YChange, 50);  fill(255, 0, 0);  circle(width - 500 + XChange, 200 + YChange, 30);    //checks if missile is true to draw the missiles on the robot  if (missile){    //missile drawings  triangle(width - 350 + XChange, 250 + missileChange, 700 + XChange, 150 + missileChange, 750 + XChange, 250 + missileChange);  triangle(width - 750 + XChange, 250 + missileChange, 300 + XChange, 150 + missileChange, 350 + XChange, 250 + missileChange);  rect(width - 350 + XChange, 250 + missileChange, 100, 200);  rect(width - 750 + XChange, 250 + missileChange, 100, 200);    }    //checks if a key is pressed  if(keyPressed){    //if the 'd' key is pressed then the robot moves to the right  if (key == 'd' || key == 'D'){    XChange += 10;  }    //if the 'a' key is pressed, the robot moves to the left  else if(key == 'a' || key == 'A'){    XChange += -10;  }    //if the 's' key is pressed, the arm moves down  else if(key == 's' || key == 'S'){    armChange = 100;    }    //if the 'w' key is pressed after the 's' key has been pressed, the arms move back up  else if(key == 'w' || key == 'W'){    if (armChange == 100){    armChange += -100;    }    //if missile is true and the 'w' key is pressed, the missiles 'fire' straight up  if(missile){    missileChange += -100;    }    }    //if the 'e' key is pressed, the missiles are created at the robot's side  else if(key == 'e' || key == 'E'){    missile = true;  textSize(32);  text("Missiles activated", 100, 100);    }    //if the 'p' key is pressed, the please boolean is activated to create the text that the user chooses to follow the mouse  else if(key == 'p' || key == 'P'){    please = true;  }    //if the 'q' key is pressed, the boolean please is deactivated  else if(key == 'q' || key == 'Q'){    please = false;    }  }    //checks if the mouse is pressed  if(mousePressed){    //if the left button is pressed,  if(mouseButton == LEFT){    fill(#C6222D);  circle(width - 550 + XChange, 150 + YChange, 25);  circle(width - 450 + XChange, 150 + YChange, 25);    fill(#FFFFFF);  circle(width - 500 + XChange, 200 + YChange, 50);  fill(#08F5FF);  circle(width - 500 + XChange, 200 + YChange, 30);    }    //if the right button is pressed  else if (mouseButton == RIGHT){    r = random(255);  g = random(255);  b = random(255);  }  }    //if the please boolean is true, then this chunk of code runs  if(please){    //checks if a key is pressed  if(keyPressed){    //if the key is R then the color value is changed for the text  if(key == 'r' || key == 'R'){    textR = random(255);  textG = random(255);  textB = random(255);    }    }    //displaying the text to follow the mouse cursor  textSize(32);  fill(textR, textG, textB);  text("we vibing bruh", mouseX, mouseY);    }  }   |  | | --- | | Written Response |   Some abstraction used by my program is the draw function. This function repeats indefinitely and allows for the robot to be displayed on screen. This code is hidden from the user during runtime by allowing the user to focus on inputs like key presses and mouse clicks, while this method handles the input through logic and math computation. This function takes the events that occur and apply it as the output area is refreshed, giving the user the ability to interact with the robot on screen. This helped manage the complexity of the program by limiting the amount of information that had to be handled over and over again to get the robot to refresh on screen as the user guided it. |

List Your Word Counts (for Gardiner only)

Response 2a (max 150): 95

Response 2b (max 200): 232

Response 2c (max 200): 209

Response 2d (max 200): 122

*A maximum of 750 words total, exclusive of program code.*