Create - High Scoring - Sample Response A

1. Video: https://youtu.be/PmFxhRgXFc8
2. Written Responses:
   a. *My program is a choose-your-own adventure game. I wrote this game in Python using the CodeSkulptor development environment to write the code. The purpose of my program is to have the user play a small choose-your-own adventure game in which they can pick several options for their path, which can either end in death, a happy life, or a return to the beginning to start all over again. The video illustrates the mechanics of the game, like the HP system, buttons, and text display. It also outlines the storyline of the game and the many paths that a player may take while playing it. In addition, the video demonstrates smaller features like the randomly selected enemy in the fight path.*
   b. *I encountered several difficulties while writing the code for this program. One of the first difficulties was setting up the code so that I could change the number of buttons and the text on them every time the user made a decision. The SimpleGUI module I used to create my game in Python only allows you to add buttons, not delete them, so to make sure the correct number of buttons is displayed at each step, I had to design the program so that it created a new frame for each separate step, instead of just being able to add or delete problems in the same window. I also had issues making sure that the HP bar updated properly even though the program was set up to check the HP value and change it if necessary every time the draw handler function ran, it didnt always work. As such, I had to create functions for increasing and decreasing the amount of HP instead of simply changing the variables within the other functions to ensure that the correct variables were being changed at the right time and that both the number on the screen and the size of the bar would change when the amount of HP changes.*
   c. def fight_beasts():
      global step4, step5, planb
      step4 = False
      step5 = True
      planb = True
      schrodinger()
      buttons(frame)

      I chose this algorithm because it is important as it handles what happens when the player in the adventure game fights beasts. When you fight beasts, first you need to change the variables that update which place you are in the game, by using True *and* False*. Then you have the fight, and then you reset the frames buttons.*

      The algorithm schrodinger() *shown below is integrated into the above algorithm. The algorithm* schrodinger() *is called to determine whether the player lives* (deathpick == False)*, with your health points decreased by 25 percent, or dies and then your health points are set to 0 and a corresponding message appears.*

      def schrodinger():
      global death,deathpick
         deathpick = random.choice(death)
         if deathpick == False:
            hp_percent(False,25)
            text_set("Return with a ", beastpick +" "+ random.choice(beast_item)+ ".")
         elif deathpick == True and (hp == 0 or hp <0): text_set(you hp_minus(hp)
      elif true and ! have,died.) (hp deathpick 0 hp ="=">0):
            hp_minus(hp)
            text_set("You have","died.")

      The algorithm buttons(frame) shown below is another integrated algorithm that resets the frame buttons using the buttons() function, which determines how many buttons the game should have next, what they should say, and what function each button should call based on the current state of the game. Both of these algorithms (schrodinger() and buttons(frame)) are noteworthy algorithms that are integrated in the main algorithm in order to make fighting beasts

possible in the adventure game

```
def buttons(frame):
  if step1 == True:
    button1 = frame.add_button("Start your journey", startpage, 50)
  if step2 == True:
    new_2_button(frame,"Take the left path", left_1,"Take the right path",right_1)
  if step3 == True and planb == False and left == False:
    new_2_button(frame,"Follow a tiny trail.",trail,"Follow a unicorn.",wiz)
  if step3 == True and planb == False and left == True:
    new_1_button(frame,"Stay for a few days.",camp)
  if step3 == True and planb == True and left == True:
    new_2_button(frame,"Explore the depths.", entercave,"Find a campsite.",camp)
  if step4 == True and planb == False and left == True:
    new_1_button(frame,"Continue on.",wiz)
  if step4 == True and planb == True and left == True:
    new_2_button(frame,"Fight the " + beastpick + ".", fight_beasts,
          "Escape in fear.",camp)
  if step4 == True and planb == False and left == False:
    new_2_button(frame,"Join the fairies in the forest.",fairies,"Return home at your
          normal size.",goback)
  if step5 == True and planb == True and deathpick == False and left == True:
    new_1_button(frame,"Return to the village with your spoils.", goback)
  if (step5 == True and planb == True and deathpick == True and left == True and hp == 0)
  or hp == 0:
    new_1_button(frame,"Start over", restart)
  if step5 == True and planb == True and deathpick == True and left == True and hp != 0:
    new_1_button(frame,"Escape",camp)
  if step6 == True and planb == True and left == False:
    new_2_button(frame,"Become the wizard's disciple.",disciple ,"Return with little
          knowledge of magic.", goback)
  if step7 == True and planb == True and left == False:
    new_1_button(frame,"Return to spread magical knowledge.",goback)
  if end == True:
    no_button(frame)
```

d.

```
def new_1_button(frame,text,function):
  frame.stop()
  frame = simplegui.create_frame('CYOA', FRAMEWIDTH,FRAMEHEIGHT)
  hp_color()
  frame.set_draw_handler(draw_handler)
  frame.set_canvas_background('white')
  frame.start()
  frame.add_button(text, function)
```

```
def new_2_button(frame,text_1,func_1,text_2,func_2):
    frame.stop()
hp_color()
    frame = simplegui.create_frame('CYOA', FRAMEWIDTH,FRAMEHEIGHT)
    frame.set_draw_handler(draw_handler)
    frame.set_canvas_background('white')
    frame.start()
    frame.add_button(text_1,func_1)
    frame.add_label("              ")
    frame.add_button(text_2,func_2)
def no_button(frame):
    frame.stop()
    frame = simplegui.create_frame('CYOA', FRAMEWIDTH,FRAMEHEIGHT)
    hp_color()
    frame.set_draw_handler(draw_handler)
    frame.set_canvas_background('white')
    frame.start()

def buttons(frame):
    if step1 == True:
        button1 = frame.add_button("Start your journey", startpage, 50)
    if step2 == True:
        new_2_button(frame,"Take the left path", left_1,"Take the right path",right_1)
    if step3 == True and planb == False and left == False:
        new_2_button(frame,"Follow a tiny trail.",trail,"Follow a unicorn.",wiz)
    if step3 == True and planb == False and left == True:
        new_1_button(frame,"Stay for a few days.",camp)
    if step3 == True and planb == True and left == True:
        new_2_button(frame,"Explore the depths.", entercave,"Find a campsite.",camp)
    if step4 == True and planb == False and left == True:
        new_1_button(frame,"Continue on.",wiz)
    if step4 == True and planb == True and left == True:
        new_2_button(frame,"Fight the " + beastpick + ".", fight_beasts,"Escape in fear.",
                     camp)
    if step4 == True and planb == False and left == False:
        new_2_button(frame,"Join the fairies in the forest.",fairies,"Return home at your
                     normal size.",goback)
    if step5 == True and planb == True and deathpick == False and left == True:
        new_1_button(frame,"Return to the village with your spoils.", goback)
    if (step5 == True and planb == True and deathpick == True and left == True and hp == 0)
            or hp == 0:
        new_1_button(frame,"Start over", restart)
    if step5 == True and planb == True and deathpick == True and left == True and hp != 0:
        new_1_button(frame,"Escape",camp)
    if step6 == True and planb == True and left == False:
        new_2_button(frame,"Become the wizard's disciple.",disciple ,"Return with little
                     knowledge of magic.", goback)
    if step7 == True and planb == True and left == False:
        new_1_button(frame,"Return to spread magical knowledge.",goback)
if end == True:
        no_button(frame)
```

This code implements a large amount of abstraction because it condenses code pieces that are essential to the program into single functions and then calls those with the function buttons. All of the buttons in this program are set by functions called new_1_button, new_2_button, or no_button that contain all of the code required to start a new

frame with that number of buttons. For each of these functions, it takes 69 lines of code to reset the frame, set its initial conditions properly, and add the new buttons with the required text. This code manages complexity well because it makes it easier to edit the code used to create the buttons because it does not need to be edited for every single condition of the buttons function. The buttons function is also an abstraction because it condenses all of the logic necessary to determine the current stage of the game into one function that can be called to check all possible cases at any point in the game where it is necessary, in addition to containing instances of abstractions. Again, this significantly condenses the program code and made it easier to create a large function because all the code to run it only needed to be edited in one place.

e.

https://secure-media.collegeboard.org/digitalServices/image/ap/create-A-code.jpg