```
////////////////////////////////////////////////////////////////
//  AUTONOMOUS PROGRAM STRATEGY CHOOSER
//  ORIGINAL CODE BY FTC TEAM# 3785, THE BESTIE BOTS
//  https://github.com/hprobotics/ftcresources/blob/master/AutonomousChooser/menu_helper.h
//  MODIFIED BY FTC TEAM# 5029, THE POWERSTACKERS
///////////////////////////////////////////////////////////////

TButtons NEXT_BUTTON = kRightButton;                                  // Create constants to make it easier to use the buttons
TButtons PREV_BUTTON = kLeftButton;
TButtons DOWN_BUTTON = kEnterButton;

/*
* Switch a boolean to the opposite value
*/
void switchBool(bool* in, TButtons activeButton){
  if(activeButton == NEXT_BUTTON || activeButton == PREV_BUTTON)      // If the active button is the left or right button:
    *in = !*in;                                                       // Toggle the input
}

/*
* Increment or decrement an integer by 1
*/
void switchInt(int* in, TButtons activeButton){
  if(activeButton == NEXT_BUTTON)                                     // If the active button is the right arrow button:
    *in = *in + 1;                                                    // Add 1 to the value
  if(activeButton == PREV_BUTTON)                                     // If the active button is the left arrow button:
    *in = *in - 1;                                                    // Subtract 1 from the value
}

/*
* Increment or decrement a floating point number by 0.1
*/
void switchFloat(float* in, TButtons activeButton){
  if(activeButton == NEXT_BUTTON)                                     // If the active button is the right arrow button:
    *in = *in + 0.1;                                                  // Add 0.1 to the value
  if(activeButton == PREV_BUTTON)                                     // If the active button is the left arrow button:
    *in = *in - 0.1;                                                  // Subtract 0.1 from the value
}
                                                                      // Options for offensive play:
bool startNear = true;                                                // Starting on the side closer to the drivers or the side f
bool doIr = true;                                                     // Placing the IR block or not
bool goAround = false;                                                  // Go around the other side of the ramp, or come back to
bool rampOtherSide = false;                                           // Go to our half of the ramp or the other alliance's half
```

```
                                                                      // Options for defensive play:
//bool blockRamp = false;                                               // Blocking the ramp or not

int delay = 0;                                                        // Delay (in seconds) applied a the start of the match
const int maxDelay = 10;                                              // Maximum possible delay

task runMenuOffensive()
{
  bDisplayDiagnostics = false;                                        // Turn off the diagnotics display
  void* currVar;                                                      // Void pointer to to store active variable
  char currType;                                                      // Identify the data type of the active variable

  currVar = &startNear;                                               // Set the current variable to startNear
  currType = 'b';                                                     // Set the current data type to boolean

  while (true){                                                       // Loop forever
    if(delay < 0)                                                     // If the delay is below 0:
      delay = 0;                                                      // Set the delay to 0
    else if(delay > maxDelay)                                         // If the delay is above the maximum:
      delay = maxDelay;                                               // Set the delay to the maximum

    nxtDisplayString(0, "Near:     %s", startNear ? "yes":"no ");     // Display all the variables and values
    nxtDisplayString(1, "Do Ir:    %s", doIr ? "yes":"no ");
    nxtDisplayString(2, "Go Around:%s", goAround ? "yes":"no ");
    nxtDisplayString(3, "RmpOthrSd:%s", rampOtherSide ? "yes":"no ");
    nxtDisplayString(4, "Delay:    %2d", delay);

    if(currVar == &startNear){                                        // Print a selection icon next to the active variable
      nxtDisplayStringAt(94, 63, "]");
      nxtDisplayStringAt(94, 55, " ");
      nxtDisplayStringAt(94, 47, " ");
      nxtDisplayStringAt(94, 39, " ");
      nxtDisplayStringAt(94, 31, " ");
    }else if(currVar == &doIr){
      nxtDisplayStringAt(94, 63, " ");
      nxtDisplayStringAt(94, 55, "]");
      nxtDisplayStringAt(94, 47, " ");
      nxtDisplayStringAt(94, 39, " ");
      nxtDisplayStringAt(94, 31, " ");
    }else if(currVar == &goAround){
      nxtDisplayStringAt(94, 63, " ");
      nxtDisplayStringAt(94, 55, " ");
      nxtDisplayStringAt(94, 47, "]");
      nxtDisplayStringAt(94, 39, " ");
      nxtDisplayStringAt(94, 31, " ");
    }else if(currVar == &rampOtherSide){
      nxtDisplayStringAt(94, 63, " ");
      nxtDisplayStringAt(94, 55, " ");
      nxtDisplayStringAt(94, 47, " ");
      nxtDisplayStringAt(94, 39, "]");
      nxtDisplayStringAt(94, 31, " ");
    }else if(currVar == &delay){
      nxtDisplayStringAt(94, 63, " ");
      nxtDisplayStringAt(94, 55, " ");
      nxtDisplayStringAt(94, 47, " ");
      nxtDisplayStringAt(94, 39, " ");
      nxtDisplayStringAt(94, 31, "]");
    }

    if(nNxtButtonPressed == NEXT_BUTTON ||                            // If the right or left arrow button is pressed:
      nNxtButtonPressed == PREV_BUTTON){
      if(currType == 'b')                                             // If the data type is boolean:
        switchBool(currVar, nNxtButtonPressed);                       // Switch the boolean variable
      else if(currType == 'i')                                        // If the data type is integer:
        switchInt(currVar, nNxtButtonPressed);                        // Switch the integer variable
      PlaySound(soundBlip);                                           // Play a sound
      ClearTimer(T1);                                                 // Clear the timer
      while(nNxtButtonPressed != kNoButton && time1[T1] <= 400){      // If any button is pressd, AND less than four seconds have
                                                                      // Do nothing
      }
    }

    if(nNxtButtonPressed == DOWN_BUTTON){                             // If the center orange button is pressed:
      if(currVar == &startNear){                                      // Set the current variable to the next in the list
        currVar = &doIr;                                              // Set the current data type to the appropriate type
        currType = 'b';
      }else if(currVar == &doIr){
        currVar = &goAround;
        currType = 'b';
      }else if(currVar == &goAround){
        currVar = &rampOtherSide;
        currType = 'b';
      }else if(currVar == &rampOtherSide){
        currVar = &delay;
        currType = 'i';
      }else if(currVar == &delay){
        currVar = &startNear;
        currType = 'b';
      }
      PlaySound(soundBlip);                                                  // Play a sound
      ClearTimer(T1);                                                        // Clear the timer
      while(nNxtButtonPressed != kNoButton && time1[T1] <= 400){             // While any button is pressed, and less than four seconds
                                                                             // Do nothing
      }
    }
  }


}

/*
* Print the selected options to the debug stream
*/
void printMenuChoices(){
  writeDebugStreamLine("Start on near side: %s\nFind IR basket: %s\nGo around far end of ramp: %s\nGo to the other half of the ra
    (startNear)? "Yes":"No", (doIr)? "Yes":"No", (goAround)? "Yes":"No", (rampOtherSide)? "Yes":"No", delay);
}
```