Create – Applications From Ideas
Written Response Submission Template

Please see Assessment Overview and Performance Task Directions for Student for the task directions and recommended word counts.

Program Purpose and Development

2a)

The program toolScript.cpp runs with C++ as the core programming language. The purpose of the program is to automate system wide tasks on a macOS/Darwin system and automate tool downloads for cybersecurity and programming jobs using homebrew. The video will illustrate two features of the program which are the ability to create new users and the ability to open any applications that are in the mac's application directory.

2b)

In order to structure the program, my mindset chose to organize the if/else layout first. I mapped out the features of the program, creating different functions that would be called upon in the main code block and from there it was easy to map out the rest of the features available. For instance, jumping from a feature to a feature within that feature was no easy feat, organizing the main block to jump to the user account alteration function and then the different segments of the function was made easier to navigate when using if/else statements for structure. Eventually, writing down a tree to wrap everything together made it easy to structure. Another issue was turning whole strings into variables for system calling in C++, c++ documentation was the best way to figure out the syntax and it eventually turned out perfectly for system automation.

2c)

```
        void usrDelete(){
  //Delete User
  printf("Type in Username to Delete\n");
  printf(":");
  scanf("%c", &endLineChar);
  cin.getline(delUser, 256);
  std::string userDelFunc("delete /Users/");
  system(("sudo dscl . " +userDelFunc
+delUser).c_str());
  //Moving user files to backup
  std::string backuppt1("/users/");
  std::string backuppt2(" /Users/Deleted\\
Users");
  system(("sudo mv " +backuppt1 +delUser
+backuppt2).c_str());
  //Delete another user
  printf("-------------------------\n");
  printf("Delete Another User? (y/n)\n");
  printf("-------------------------\n");
  printf(":");
  scanf("%s", usrDelFunc);
}
//Delete User
if(!strcmp(options, "b") || !strcmp(options,
"B")){
  //Enable root and delete user
  printf("Enabling Root...\n");
  printf("Enter Name of Root Admin...\n");
  system("dsenableroot");
  //Delete User Function
  usrDelete();
```

```
    //Conditional Loop to continue or
cancel delete
    if(!strcmp(usrDelFunc, "y") ||
!strcmp(usrDelFunc, "Y")){
        while (true){
            usrDelete();
            if(!strcmp(usrDelFunc, "n") ||
!strcmp(usrDelFunc, "N")){
                break;
            }
        }
    }
    //Disable Root
    printf("Disabliing root user (Re-enter root
admin name)\n");
    system("dsenableroot -d");
    printf("Returning to menu...\n");
    sleep(1);
    initializationApp();
}
```

This code block is used to automate deleting users off a macOS/Darwin system.
The “void usrDelete();” function serves as the algorithm that performs system calls to delete the user, backup its files in a separate directory, and then gives another option to delete another user. The “//Delete User” is the algorithm that gives root privileges to actually delete the user. After enabling root, it calls upon the “usrDelete();” function, then utilizes a conditional loop to either continue deleting more users or disable the root privileges and returns to the main program block, depending on whether or not the user chooses to continue deleting users or leave.

2d)

```
        //Adding unique ID to User
    printf("Creating Unique ID...\n");
    int numID = rand() % 9000 + 1000;
```

```
   uniqueID = numID;
   printf("Unique ID = %d\n", uniqueID);
   std::string sudoIDpt1("-create /Users/");
   std::string sudoIDpt2(" UniqueID ");
   std::string sudoIDpt3(""
+std::to_string(uniqueID));
   system(("sudo dscl . " +sudoIDpt1
+newUser +sudoIDpt2 +sudoIDpt3).c_str());
```

My string variable system is what allowed me to take numerical data and use it as a variable in a system call. Without the variables like “numID” or “sudoID,” it would not have been possible for me to condense user inputted data into readable and easy-to-reuse variables. Looking at the “//Adding unique ID to User” code block, this segment creates the unique user IDs that a macOS/Darwin system needs to create new user accounts. To avoid any confusion and issues with random number generation to create a unique ID, my abstraction condenses the number generation into one int variable that then converts into a string, which therefore becomes usable in a C++ system call.

3) On a separate file called “Written Response Question 3”