A black background with purple text

AI-generated content may be incorrect.

**FUNDEMANTALS OF ALGORITHM & COMPUTER PROBLEM SOLVING (CSC126)**

**SEMESTER MAC-OGOS 2024/2025**

PROJECT REPORT:

**Go-Kart Booking System**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Group Members Details | |  |  |
|  | Group Members | Student No. | Group | Program |
| 1 | IRFAN SHAH BIN MAIZUL HISHAM | 2025171523 | RCDCS1101A | CDSC110 |
| 2 | SHAHRIN AREFF SHAH BIN SHAH RIZAL | 2025151503 | RCDCS1101A | CDSC110 |
| 3 | MUHAMMAD BIN ABDUL AZIZ | 2025507823 | RCDCS1101B | CDSC110 |
|  | Telephone Number (Leader): | 0196093161 | | |

Lecturer’s Name: Nora Yanti Binti Che Jan

**TABLE OF CONTENTS**

1. PROJECT SUMMARY
2. OBJECTIVE OF THE PROJECT
3. ALRGORITHM DESIGN
4. CODING GUIDELINES
5. SOURCE CODE
6. SAMPLES OF INPUT AND OUTPUT
7. DISCUSSIONS
8. CONLCUSION
9. PRESENTATION [INSERT LINK HERE]
10. **Project Summary**

The project is a terminal based Go-Kart Booking System that is coded in C++.

1. **OBJECTIVES OF THE PROJECT**
2. To implement and showcase every controlled structure learned in CSC126 suitably based on the problem.
3. To create a system which trakcs and allows for customers to do booking.
4. To calculate the cost of booking accordingly based on factors such as driver count and engine capacity
5. **ALGORITHM DESIGN**
   1. **Flowcharts**

|  |
| --- |
|  |

* 1. **Pseudocode**

|  |
| --- |
| FUNCTION INTEGER main()  CALL menu()  DECLARE CHARACTER continueChoice  DECLARE INTEGER customer = 0  DECLARE FLOAT totalIncome = 0  DO  //Initializations (in order)  STRING bookingType  DECLARE INTEGER driverCount  DECLARE STRING driverName[5]  DECLARE INTEGER driverAge[5]  DECLARE CHARACTER license[5]  DECLARE STRING raceFormat  DECLARE STRING track    DECLARE INTEGER engineCapacity[5]  DECLARE INTEGER laps  DECLARE STRING helmetSize[5], suitSize[5]  DECLARE INTEGER shoeSize[5]  DECLARE FLOAT helmetPrice[5], suitPrice[5], shoePrice[5]  DECLARE FLOAT gearPrice[5]  // The Crux of the Go-Kart Booking System  SET bookingType = getBookingType()    SET driverCount = getDriverCount(bookingType)  CALL getDriverDetails(driverCount, driverName, driverAge, license)  SET raceFormat = setRaceFormat(bookingType)  SET track = setTrack(bookingType, raceFormat)  SET setEngineCapacity(driverCount, driverName, driverAge, license, engineCapacity)  SET laps = setLaps(driverCount, raceFormat)  FOR INTEGER i = 0; i < driverCount; i++  CALL driverGear(i, driverName)  CALL selectHelmet(i,helmetSize, helmetPrice)  CALL selectSuit(i,suitSize, suitPrice)  CALL selectShoe(i, shoeSize, shoePrice)  END FOR  DECLARE FLOAT totalGearPrice  SET totalGearPrice = calculateTotalGearPrice(driverCount, helmetPrice, suitPrice, shoePrice, gearPrice)  DECLARE FLOAT membershipDiscount  SET membershipDiscount = setMembershipDiscount()  DECLARE FLOAT totalPrice  SET totalPrice = calculatePrice(engineCapacity, laps, driverCount, totalGearPrice, membershipDiscount)  OUTPUT "Race Format: ", raceFormat  OUTPUT "Track: ", track  std::cout << std::fixed << std::showpoint;  FOR INTEGER i = 0; i < driverCount; i++  OUTPUT "Driver: ", driverName[i]  OUTPUT "Age: ", driverAge[i]  OUTPUT "Has License: ", license[i]  OUTPUT "Engine Capacity: ", engineCapacity[i], "cc"  OUTPUT "Helmet Size: ", helmetSize[i]  OUTPUT "Helmet Price: RM", std::setprecision(2), helmetPrice[i]  OUTPUT "Suit Size: ", suitSize[i]  OUTPUT "Suit Pricet: RM", std::setprecision(2), suitPrice[i]  OUTPUT "Shoe Size: ", shoeSize[i], "cm"  OUTPUT "Shoe Price: RM", std::setprecision(2), shoePrice[i]  OUTPUT "Gear Price: RM", std::setprecision(2), gearPrice[i]  END FOR    OUTPUT "Total Gear Price: RM", std::setprecision(2), totalGearPrice  OUTPUT "Membership Discount\t: ", membershipDiscount \* 100, "%"  OUTPUT "Total : RM", std::setprecision(2), totalPrice  totalIncome += totalPrice  customer++  PROMPT "Continue for another customer? (Y/N): "  SET continueChoice = getValidCharacterInput()  WHILE std::toupper(continueChoice) == 'Y'  OUTPUT "Total Income: RM ", std::setprecision(2), totalIncome  RETURN 0  END FUNCTION  MODULE menu()  DECLARE INTEGER menu    DO  CALL displayMenu()  PROMPT "Choose your option: "  SET menu = getValidIntegerInput("option", 0, 5)  IF menu == 2 THEN  displayRaceFormat("Group")  END IF  IF menu == 3 THEN  displayTrackList()  END IF  IF menu == 4 THEN  displayRacingGear()  END IF  IF menu == 5 THEN  displayEngineCapacities(18, 'Y')  END IF  WHILE menu != 1 && menu != 0  END MODULE  FUNCTION INTEGER getValidIntegerInput(STRING inputType, INTEGER minValue, INTEGER maxValue)  int input;  READ input;  WHILE std::cin.fail() || (input < minValue || input > maxValue)  PROMPT "\t\t\tPlease enter a valid " << inputType << " (" << minValue << " - " << maxValue << "): "  READ input  END WHILE  RETURN input  END FUNCTION  FUNCTION CHARACTER getValidCharacterInput()  STRING input  READ input  WHILE input.length() != 1 || !std::isalpha(input[0])  PROMPT "Please enter a valid input (Y/N): "  READ input  END WHILE  RETURN std::toupper(input[0])  END FUNCTION  STRING getBookingType()  DECLARE INTEGER bookingTypeID  DISPLAY "1 - Solo"  DISPLAY "2 - Group (Maximum 5)"  PROMPT "Please choose your desired booking (1 - 2): "  SET bookingTypeID = getValidIntegerInput("booking type", 1, 2)  IF bookingTypeID == 1 THEN  RETURN "Solo"  ELSE  RETURN "Group"  END IF  END FUNCTION  FUNCTION INTEGER getDriverCount(STRING& bookingType)  IF bookingType == "Solo" THEN  RETURN 1  END IF  PROMPT "Please enter the number of drivers (2 - 5): "  RETURN getValidIntegerInput("number of drivers", 2, 5)  END FUNCTION  MODULE getDriverDetails(INTEGER& driverCount, STRING driverName[], INTEGER driverAge[], CHARACTER license[])  FOR int i = 0; i < driverCount; i++  PROMPT "Driver #", (i+1), " Name: "  READ driverName[i]  PROMPT "Driver #", (i+1), " Age: "  READ driverAge[i]  PROMPT "Does Driver #", (i+1), " has a license? (Y/N): "  SET license[i] = getValidCharacterInput()  END FOR  END MODULE  FUNCTION STRING setRaceFormat(STRING& bookingType)  DECLARE INTEGER raceFormatID  DECLARE STRING raceFormat  CALL displayRaceFormat(bookingType)  IF bookingType == "Solo" THEN  PROMPT "Please choose the race format (1 - 4): "  SET raceFormatID = getValidIntegerInput("race format", 1, 4);  ELSE IF bookingType == "Group" THEN  PROMPT "Please choose the race format (1 - 5): "  SET raceFormatID = getValidIntegerInput("race format", 1, 5)  END IF  switch (raceFormatID)  case 1: return "Circuit Race";  case 2: return "Sprint Race";  case 3: return "Time Trial";  case 4: return "Drag Race";  case 5: return "Elimination Race";  default: return "Invalid Race Format";  END FUNCTION  FUNCTION STRING setTrack(STRING& bookingType, STRING& raceFormat)  DECLARE INTEGER trackID  IF bookingType == "Group" AND raceFormat == "Circuit Race" THEN  DISPLAY "Available Track:"  DISPLAY "1 - Section 9 Circuit"  DISPLAY "2 - Blackrock Circuit"  PROMPT "Please choose your track (1 - 2): "  SET trackID = getValidIntegerInput("track, 1, 2)  ELSE IF raceFormat == "Time Trial" THEN  DISPLAY "Available Track:"  DISPLAY "1 - Section 9 Circuit"  DISPLAY "2 - Blackrock Circuit"  DISPLAY "3 - Rushline Dash"  DISPLAY "4 - Chrono Pass"  PROMPT "Please choose your track (1 - 4): "  SET trackID = getValidIntegerInput("track, 1, 4)  END IF  switch (trackID)  case 1: return "Section 9 Circuit";  case 2: return "Blackrock Circuit";  case 3: return "Rushline Dash";  default: return "Chrono Pass";  IF bookingType == "Solo" AND raceFormat == "Circuit Race" THEN  DISPLAY "Available Track: Section 9 Circuit"  DISPLAY "Defaulting to said track"  RETURN "Section 9 Circuit"  ELSE IF raceFormat == "Elimination Race" THEN  DISPLAY "Available Track: Blackrock Circuit"  DISPLAY "Defaulting to said track"  RETURN "Blackrock Circuit"  ELSE IF raceFormat == "Sprint Race" THEN  DISPLAY "Available Track: Rushline Dash"  DISPLAY "Defaulting to said track"  RETURN "Rushline Dash"  ELSE IF raceFormat == "Drag Race" THEN  DISPLAY "Available Track: Torque Strip"  DISPLAY "Defaulting to said track"  RETURN "Torque Strip"  END IF  RETURN "Track"  END FUNCTION    MODULE setEngineCapacity(INTEGER driverCount, STRING driverName[], INTEGER driverAge[], CHARACTER license[])  FOR INTEGER i = 0, i < driverCount, i++  DISPLAY "Driver : ", driverName[i]  CALL displayEngineCapacities(driverAge[i], license[i])  IF driverAge[i] < 13 THEN  DISPLAY "Driver under 13: Go-Kart must be under 200cc"  DISPLAY "Defaulting to 120cc"  SET engineCapacity[i] = 120  CONTINUE  END IF  IF license[i] != 'Y' THEN  DISPLAY "Choose your desired engine capacity (1 - 2): "  SET engineCapacity[i] = getValidIntegerInput("engine capacity", 1, 2)  END IF  IF license[i] == 'Y'  DISPLAY "Choose your desired engine capacity (1 - 3): "  SET engineCapacity[i] = getValidIntegerInput("engine capacity", 1, 3)  END IF  switch (engineCapacity[i])  case 1: engineCapacity[i] = 120; break;  case 2: engineCapacity[i] = 200; break;  default: engineCapacity[i] = 270; break;  END FOR  END MODULE  FUNCTION setLaps(INTEGER driverCount, STRING& raceFormat)  DECLARE INTEGER laps  IF raceFormat == "Circuit Race" THEN  DISPLAY "Available number of laps: 2, 3, 4"  PROMPT "How many laps would you like? (2/3/4): "  READ laps  ELSE IF raceFormat == "Sprint Race" OR raceFormat == "Drag Race" THEN  DISPLAY "Sprint Races and Drag Races only have 1 lap"  SET laps = 1  ELSE IF raceFormat == "Time Trial" THEN  DISPLAY "Available number of laps: Unlimited"  PROMPT "How many laps would you like? :"  READ laps  ELSE  DISPLAY "Number of laps corresponds with the number of drivers"  SET laps = driverCount - 1  DISPLAY "Laps = Driver Count - 1"  DISPLAY "Laps = ", driverCount, " - 1"  DISPLAY "Laps = ", laps  END IF  RETURN LAPS  END FUNCTION  MODULE driverGear(INTEGER& i, STRING driverName[])  displayRacingGear()  DISPLAY "Driver: ", driverName[i]  END MODULE  FUNCTION float selectHelmet (INTEGER i, STRING helmetSize[], FLOAT helmetPrice[])  PROMPT "Please choose your helmet size (S/M/L/XL): "  READ helmetSize[i]  SET transform(helmetSize[i].begin(), helmetSize[i].end(), helmetSize[i].begin(), ::toupper)  WHILE cin.fail() || (helmetSize[i] != "S" && helmetSize[i] != "M" && helmetSize[i] != "L" && helmetSize[i] != "XL")  PROMPT "Please choose a valid helmet size (S/M/X/XL): "  READ helmetSize[i]  SET transform(helmetSize[i].begin(), helmetSize[i].end(), helmetSize[i].begin(), ::toupper)  END WHILE  IF helmetSize[i] == "S" THEN  helmetPrice[i] = 2  END IF  IF helmetSize[i] == "M" THEN  helmetPrice[i] = 4  END IF  IF helmetSize[i] == "L" THEN  helmetPrice[i] = 6  END IF  IF helmetSize[i] == "XL" THEN  helmetPrice[i] = 8  END IF  RETURN helmetPrice[i]  END FUNCTION  FUNCTION float selectSuit (INTEGER i, STRING suitSize[], FLOAT suitPrice[])  PROMPT "Please choose your suit size (S/M/L/XL): "  READ suitSize[i]  SET std::transform(suitSize[i].begin(), suitSize[i].end(), suitSize[i].begin(), ::toupper)  WHILE cin.fail() || (suitSize[i] != "S" && suitSize[i] != "M" && suitSize[i] != "L" && suitSize[i] != "XL")  PROMPT "Please choose a valid helmet size (S/M/X/XL): "  READ helmetSize[i]  SET std::transform(suitSize[i].begin(), suitSize[i].end(), suitSize[i].begin(), ::toupper)  END WHILE  IF suitSize[i] == "S" THEN  suitPrice[i] = 2  END IF  IF suitSize[i] == "M" THEN  suitPrice[i] = 4  END IF  IF suitSize[i] == "L" THEN  suitPrice[i] = 6  END IF  IF helmetSize[i] == "XL" THEN  suitPrice[i] = 8  END IF  RETURN suitPrice[i]  END FUNCTION  FUNCTION FLOAT selectShoe(INTEGER i, INTEGER shoeSize[], FLOAT shoePrice[])  PROMPT "Please choose your shoe size (20 - 35): "  READ shoeSize[i]  WHILE std::cin.fail() || (shoeSize[i] < 20 || shoeSizep[i] > 35  PROMPT "Please choose a valid shoe size (20 - 35): "  READ shoeSize[i]  END WHILE  IF shoeSize[i] <= 25 THEN  shoePrice[i] = 3  ELSE  shoePrice[i] = 6  END IF  RETURN shoePrice[i]  END FUNCTION  FUNCTION FLOAT calculateTotalGearPrice(INTEGER& driverCount, FLOAT helmetPrice[], FLOAT suitPrice[], FLOAT gearPrice[])  DECLARE totalGearPrice  SET totalGearPrice = 0  FOR INTEGER i = 0, i < driverCount, i++  gearPrice[i] = helmetPrice[i] + suitPrice[i] + shoePrice [i]  totalGearPrice += gearPrice[i]  END FOR  RETURN totalGearPrice  END FUNCTION  FUNCTION FLOAT setMembershipDiscount ()  DECLARE CHAR membership  PROMPT "Do you have a membership? (Y/N): "  SET membership = getValidCharacterInput()  IF membership == 'Y' THEN  RETURN 0.1  ELSE  RETURN 0.0  END IF  END FUNCTION  FUNCTION FLOAT calculatePrice (int engineCapacity[], int laps, int driverCount, float totalGearPrice, float membershipDiscount)  SET pricePerCC = 03  SET totalKartPrice = 0  FOR i = 0, i < driverCount, i++  totalKartPrice += (engineCapacity[i] \* pricePerCC) \* laps  END FOR    finalPrice = (totalKartPrice + totalGearPrice) \* (1 - membershipDiscount)  END FUNCTION |

1. **CODING GUIDELINES**
2. cameCase naming convention

This is a popular naming convention and it is used by our lecturer to teach C++.

1. Tab Width: 8

The usage of 8 tab width is to easily spot excessive nesting easily which ties to the next point.

1. Maximum Level of Indentation: 3

To quote Linus Torvalds on the Linux Kernal coding guideline: “If you need more than 3 levels of indentation, your code is broken anyway and should fix it”.

1. Indentation Style: Stroustrup

Bajrne Stroustrup is the inventor of C++ and his style of coding is in the C++ International Standard Organization (ISO).

1. No *using namespace std*

The usage of *using namespace std* in simpler programs is okay and sometimes advised to but at higher levels of programming, the usage of it can cause problems. To avoid the compiler being confused and to better make the usage of functions from the standard library clearer, *using namesapce std* is omitted from our code.

1. No abbreviations

Abbreviations can make coding easier, but with the advent of IDE’s autocomplete, coding takes less keystroke than ever. Thus, to make the code readable to everyone, it does not matter if you’re working on it or just an outsider, the code will still be comprehensible even if you have no idea on Go-Kart or how this booking works.

1. **SOURCE CODE**

|  |
| --- |
| // CSC126 Group Project  // Go-Kart Booking System  // IRFAN SHAH BIN MAIZUL HISHAM                 (2025171523) (shahxvi)  // SHAHRIN AREFF SHAH BIN SHAH RIZAL     (2025151503) (rinnnnnn17)  // MUHAMMAD BIN ABDUL AZIZ                      (2025507823) (Kundoo)  #include <iostream>  #include <string>  #include <cmath>  #include <iomanip>  #include <cctype>  #include <algorithm>  void menu();  void displayMenu();  void displayRaceFormat(std::string bookingType);  void displayTrackList();  void displayRacingGear();  void displayEngineCapacities(int age, char license);  int getValidIntegerInput (std::string inputType, int minValue, int maxValue);  char getValidCharacterInput();  std::string getBookingType();  int getDriverCount(std::string& bookingType);  void getDriverDetails(int& driverCount, std::string driverName[], int driverAge[], char license[]);  std::string setRaceFormat(std::string& bookingType);  std::string setTrack(std::string& bookingType, std::string& raceFormat);  void setEngineCapacity(int driverCount, std::string driverName[], int driverAge[], char license[], int engineCapacity[]);  int setLaps(int driverCount, std::string& raceFormat);  void driverGear(int& i, std::string driverName[]);  float selectHelmet(int i, std::string helmetSize[], float helmetPrice[]);  float selectSuit(int i, std::string suitSize[], float suitPrice[]);  float selectShoe(int i, int shoeSize[], float shoePrice[]);  float calculateTotalGearPrice(int& driverCount, float helmetPrice[], float suitPrice[], float shoePrice[], float gearPrice[]);  float setMembershipDiscount();  float calculatePrice(int engineCapacity[], int laps, int driverCount, float gearPrice, float membershipDiscount);  int main()  {          menu();          char continueChoice;          int customer = 0;          float totalIncome = 0;          do {                  //Initializations (in order)                  std::string bookingType;                  int driverCount;                  std::string driverName[5];                  int driverAge[5];                  char license[5];                  std::string raceFormat;                  std::string track;                    int engineCapacity[5];                  int laps;                  std::string helmetSize[5], suitSize[5]; int shoeSize[5];                  float helmetPrice[5], suitPrice[5], shoePrice[5];                  float gearPrice[5];                  // The Crux of the Go-Kart Booking System                  bookingType = getBookingType();                    driverCount = getDriverCount(bookingType);                  getDriverDetails(driverCount, driverName, driverAge, license);                  raceFormat = setRaceFormat(bookingType);                  track = setTrack(bookingType, raceFormat);                  setEngineCapacity(driverCount, driverName, driverAge, license, engineCapacity);                  laps = setLaps(driverCount, raceFormat);                  for (int i = 0; i < driverCount; i++) {                          driverGear(i, driverName);                          selectHelmet(i,helmetSize, helmetPrice);                          selectSuit(i,suitSize, suitPrice);                          selectShoe(i, shoeSize, shoePrice);                  }                  float totalGearPrice = calculateTotalGearPrice(driverCount, helmetPrice, suitPrice, shoePrice, gearPrice);                  float membershipDiscount = setMembershipDiscount();                  float totalPrice = calculatePrice(engineCapacity, laps, driverCount, totalGearPrice, membershipDiscount);                  system("cls");                  std::cout << "\n\t\t\tRace Format: " << raceFormat                            << "\n\t\t\tTrack: " << track << "\n";                  std::cout << std::fixed << std::showpoint;                  for (int i = 0; i < driverCount; i++) {                          std::cout << "\n\t\t\tDriver\t\t\t: " << driverName[i]                                    << "\n\t\t\tAge\t\t\t: " << driverAge[i]                                    << "\n\t\t\tHas License\t\t: " << license[i]                                    << "\n\t\t\tEngine Capacity\t\t: " << engineCapacity[i] << "cc"                                    << "\n\t\t\tHelmet Size\t\t: " << helmetSize[i]                                    << "\n\t\t\tHelmet Price\t\t: RM" <<  std::setprecision(2) << helmetPrice[i]                                    << "\n\t\t\tSuit Size\t\t: " << suitSize[i]                                    << "\n\t\t\tSuit Price\t\t: RM" <<  std::setprecision(2) << suitPrice[i]                                    << "\n\t\t\tShoe Size\t\t: " << shoeSize[i] << "cm"                                    << "\n\t\t\tShoe Price\t\t: RM" << std::setprecision(2) << shoePrice[i]                                    << "\n\t\t\tGear Price\t\t: RM" <<  std::setprecision(2) << gearPrice[i] << "\n";                  }                    std::cout << "\n\t\t\tTotal Gear Price: RM" <<  std::setprecision(2) << totalGearPrice                            << "\n\t\t\tMembership Discount\t: " << membershipDiscount \* 100 << "%"                            << "\n\t\t\tTotal : RM" <<  std::setprecision(2) << totalPrice;                  totalIncome += totalPrice;                  customer++;                  std::cout << "\n\t\t\tContinue for another customer? (Y/N): ";                  continueChoice = getValidCharacterInput();                  system("cls");          } while (std::toupper(continueChoice) == 'Y');          std::cout << "\n\t\t\tTotal Income: RM " <<  std::setprecision(2) << totalIncome << "\n";            system("pause");          return 0;  }  void menu()  {          int menu;            do {                  displayMenu();                  std::cout << "\n\t\t\tChoose your option: ";                  menu = getValidIntegerInput("option", 0, 5);                  if (menu == 2) {                          system("cls");                          displayRaceFormat("Group");     // Parameters to allow the menu to show every option                  }                  if (menu == 3) {                          system("cls");                          displayTrackList();                  }                  if (menu == 4) {                          system("cls");                          displayRacingGear();                  }                  if (menu == 5) {                          system("cls");                          displayEngineCapacities(18, 'Y');       // Parameters to allow the menu to show every option                  }          } while (menu != 1 && menu != 0);  }  void displayMenu()  {          std::cout << "\t\_\_\_\_\_\_\_\_\_            \_\_\_\_\_\_ \_\_             \_\_\_\_\_  \_\_\_\_\_\_\_\_            \_\_\_\_\_\_ \_\_\_\_\_                 \_\_\_\_\_\_\_\_              \_\_\_\_\_                 "                    << "\n\t\_\_  \_\_\_\_/\_\_\_\_\_       \_\_\_  //\_/\_\_\_\_\_ \_\_\_\_\_\_\_\_\_  /\_ \_\_\_  \_\_ )\_\_\_\_\_\_\_\_\_\_\_\_\_\_  /\_\_\_\_(\_)\_\_\_\_\_\_\_\_\_\_\_\_\_ \_ \_\_  \_\_\_/\_\_\_\_  \_\_\_\_\_\_\_\_\_\_  /\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_ "                    << "\n\t\_  / \_\_ \_  \_\_ \\\_\_\_\_\_\_\_\_  ,<  \_  \_\_ `/\_  \_\_\_/  \_\_/ \_\_  \_\_  |  \_\_ \\  \_\_ \\\_  //\_/\_  /\_\_  \_\_ \\\_  \_\_ `/ \_\_\_\_\_ \\\_\_  / / /\_  \_\_\_/  \_\_/  \_ \\\_  \_\_ `\_\_ \\"                    << "\n\t/ /\_/ / / /\_/ //\_\_\_\_\_/  /| | / /\_/ /\_  /   / /\_   \_  /\_/ // /\_/ / /\_/ /  ,<  \_  / \_  / / /  /\_/ /  \_\_\_\_/ /\_  /\_/ /\_(\_\_  )/ /\_ /  \_\_/  / / / / /"                    << "\n\t\\\_\_\_\_/  \\\_\_\_\_/       /\_/ |\_| \\\_\_,\_/ /\_/    \\\_\_/   /\_\_\_\_\_/ \\\_\_\_\_/\\\_\_\_\_//\_/|\_| /\_/  /\_/ /\_/\_\\\_\_, /   /\_\_\_\_/ \_\\\_\_, / /\_\_\_\_/ \\\_\_/ \\\_\_\_//\_/ /\_/ /\_/ "                    << "\n\t                                                                                         /\_\_\_\_/           /\_\_\_\_/                               "                    << "\n\t\t\t1 - Start Booking"                    << "\n\t\t\t2 - Race Formats"                    << "\n\t\t\t3 - Track Lists"                    << "\n\t\t\t4 - Racing Gear Sizes & Prices"                    << "\n\t\t\t5 - Go-Kart Engine Capacities"                    << "\n\t\t\t0 - Exit\n";  }  void displayRaceFormat(std::string bookingType)  {          std::cout << "\n\t\t\tAvailable Race Formats:"                            << "\n\t\t\t1 - Circuit Race"                            << "\n\t\t\t2 - Sprint Race"                            << "\n\t\t\t3 - Time Trial"                            << "\n\t\t\t4 - Drag Race";          if (bookingType == "Group") {                  std::cout << "\n\t\t\t5 - Eliminationn Race (Group)\n";          }  }  void displayTrackList()  {          std::cout << "\n\t\t\tAvailable Tracks:"                    << "\n\t\t\t1 - Section 9 Circuit"                    << "\n\t\t\t2 - Chrono Pass"                    << "\n\t\t\t3 - Rushline Dash"                    << "\n\t\t\t4 - Blackrock Circuit"                    << "\n\t\t\t5 - Torque Strip\n";  }  void displayRacingGear()  {          std::cout << "\n\t\t\tRacing Gears:"                            << "\n\t\t\t\t\t\tHelmets\t\t\tSuits\t\t\tShoes\n"                            << "\n\t\t\tSize S:\t\t\tRM 2\t\t\tRM 5\t\t\tSize 20-25 cm: RM 3"                            << "\n\t\t\tSize M:\t\t\tRM 4\t\t\tRM 10\t\t\tSize 26-35 cm: RM 6"                            << "\n\t\t\tSize L:\t\t\tRM 6\t\t\tRM 15"                            << "\n\t\t\tSize XL\t\t\tRM 8\t\t\tRM 20\n";  }  void displayEngineCapacities(int age, char license)  {          std::cout << "\n\t\t\tAvailable Engine Capacities:";          std::cout << "\n\t\t\t1 - 100cc (Under 13)";          if (age >= 18) {                  std::cout << "\n\t\t\t2 - 200cc";                  if (license == 'Y')                          std::cout << "\n\t\t\t3 - 270cc (Requires License)\n";          }  }  int getValidIntegerInput(std::string inputType, int minValue, int maxValue)  {          int input;          std::cin >> input;          while (std::cin.fail() || (input < minValue || input > maxValue)) {                  std::cin.clear();                  std::cin.ignore(1000, '\n');                  std::cout << "\t\t\tPlease enter a valid " << inputType << " (" << minValue << " - " << maxValue << "): ";                  std::cin >> input;          }          return input;  }  char getValidCharacterInput()  {          std::string input;          std::cin >> input;          while (input.length() != 1 || !std::isalpha(input[0])) {                  std::cin.clear();                  std::cin.ignore(1000, '\n');                  std::cout << "\t\t\tPlease enter a valid input (Y/N): ";                  std::cin >> input;          }          return std::toupper(input[0]);  }  std::string getBookingType()  {          int bookingTypeID;          std::cout << "\n\t\t\t1 - Solo\n"                    << "\t\t\t2 - Group (Maximum 5)\n"                    << "\t\t\tPlease choose your desired booking (1 - 2): ";          bookingTypeID = getValidIntegerInput("booking type", 1, 2);          if (bookingTypeID == 1) {                  return "Solo";          }          else {                  return "Group";          }  }  int getDriverCount(std::string& bookingType)  {          if (bookingType == "Solo")                  return 1;          std::cout << "\t\t\tPlease enter the number of drivers (2 - 5): ";          return getValidIntegerInput("number of drivers", 2, 5);  }  void getDriverDetails(int& driverCount, std::string driverName[], int driverAge[], char license[])  {          for (int i = 0; i < driverCount; i++) {                  std::cin.ignore();                  std::cout << "\n\t\t\tDriver #" << (i+1) << " Name: ";                  std::getline(std::cin, driverName[i]);                  std::cout << "\t\t\tDriver #" << (i+1) << " Age: ";                  std::cin >> driverAge[i];                  std::cout << "\t\t\tDoes Driver #" << (i+1) << " has a license? (Y/N): ";                  license[i] = getValidCharacterInput();          }  }  std::string setRaceFormat(std::string& bookingType)  {          int raceFormatID;          std::string raceFormat;          displayRaceFormat(bookingType);          if (bookingType == "Solo") {                  std::cout << "\n\t\t\tPlease choose the race format (1 - 4): ";                  raceFormatID = getValidIntegerInput("race format", 1, 4);          }          else if (bookingType == "Group") {                  std::cout << "\n\t\t\tPlease choose the race format (1 - 5): ";                  raceFormatID = getValidIntegerInput("race format", 1, 5);          }            switch (raceFormatID) {                  case 1: return "Circuit Race";                  case 2: return "Sprint Race";                  case 3: return "Time Trial";                  case 4: return "Drag Race";                  case 5: return "Elimination Race";                  default: return "Invalid Race Format";          }  }  std::string setTrack(std::string& bookingType, std::string& raceFormat)  {          int trackID;          if (bookingType == "Group" && raceFormat == "Circuit Race") {                  std::cout << "\n\t\t\tAvailable Track:"                            << "\n\t\t\t1 - Section 9 Circuit"                            << "\n\t\t\t2 - Blackrock Circuit"                            << "\n\t\t\tPlease choose your track (1 - 2): ";                  trackID = getValidIntegerInput("track", 1, 2);          }          else if (raceFormat == "Time Trial") {                  std::cout << "\n\t\t\tAvailable Track:"                            << "\n\t\t\t1 - Section 9 Circuit"                            << "\n\t\t\t2 - Blackrock Circuit"                            << "\n\t\t\t3 - Rushline Dash"                            << "\n\t\t\t4 - Chrono Pass"                            << "\n\t\t\tPlease choose your track (1 - 4): ";                  trackID = getValidIntegerInput("track", 1, 4);          }          switch (trackID) {                  case 1: return "Section 9 Circuit";                  case 2: return "Blackrock Circuit";                  case 3: return "Rushline Dash";                  default: return "Chrono Pass";          }          if (bookingType == "Solo" && raceFormat == "Circuit Race") {                  std::cout << "\n\t\t\tAvailable Track: Section 9 Circuit"                            << "\n\t\t\tDefaulting to said track\n";                  return "Section 9 Circuit";          }          else if (raceFormat == "Elimination Race") {                  std::cout << "\n\t\t\tAvailable Track: Blackrock Circuit"                            << "\n\t\t\tDefaulting to said track\n";                  return "Blackrock Circuit";          }          else if (raceFormat == "Sprint Race") {                  std::cout << "\n\t\t\tAvailable Track: Rushline Dash"                            << "\n\t\t\tDefaulting to said track\n";                  return "Rushline Dash";          }          else if (raceFormat == "Drag Race") {                  std::cout << "\n\t\t\tAvailable Track: Torque Strip"                            << "\n\t\t\tDefaulting to said track\n";                  return "Torque Strip";          }          return "Track"; // Should never get to this point  }  void setEngineCapacity(int driverCount, std::string driverName[], int driverAge[], char license[], int engineCapacity[])  {          for (int i = 0; i < driverCount; i++) {                  std::cout << "\n\t\t\tDriver : " << driverName[i];                  displayEngineCapacities(driverAge[i], license[i]);                  if (driverAge[i] < 13) {                          std::cout << "\n\t\t\tDriver under 13: Go-Kart must be under 200cc"                                    << "\n\t\t\tDefaulting to 120cc\n";                          engineCapacity[i] = 120;                          continue;                  }                  // Over 13 without License                  if (license[i] != 'Y') {                          std::cout << "\n\t\t\tChoose your desired engine capacity (1 - 2): ";                          engineCapacity[i] = getValidIntegerInput("engine capacity", 1 , 2);                  }                  // Over 13 with License                  if (license[i] == 'Y') {                  std::cout << "\n\t\t\tChoose your desired engine capacity (1 - 3): ";                  engineCapacity[i] = getValidIntegerInput("engine capacity", 1 , 3);                  }                  switch (engineCapacity[i]) {                          case 1: engineCapacity[i] = 120; break;                          case 2: engineCapacity[i] = 200; break;                          default: engineCapacity[i] = 270; break;                  }          }  }  int setLaps(int driverCount, std::string& raceFormat) {          int laps;          if (raceFormat == "Circuit Race") {                  std::cout << "\n\t\t\tAvailable number of laps: 2, 3, 4"                            << "\n\t\t\tHow many laps would you like? (2/3/4): ";                  std::cin >> laps;          }          else if (raceFormat == "Sprint Race" || raceFormat == "Drag Race") {                  std::cout << "\n\t\t\tSprint Races and Drag Races only have 1 lap\n";                  laps = 1;          }          else if (raceFormat == "Time Trial") {                  std::cout << "\n\t\t\tAvailable number of laps: Unlimited"                            << "\n\t\t\tHow many laps would you like?: ";                  std::cin >> laps;          }          else { // Elimination Race                  std::cout << "\n\t\t\tNumber of laps corresponds with the number of drivers";                  laps = driverCount - 1;                  std::cout << "\n\t\t\tLaps = Driver Count - 1"                            << "\n\t\t\tLaps = " << driverCount << " - 1"                            << "\n\t\t\tLaps = " << laps << "\n";          }          return laps;  }  void driverGear(int& i, std::string driverName[])  {          displayRacingGear();          std::cout << "\n\t\t\tDriver: " << driverName[i];  }  float selectHelmet(int i, std::string helmetSize[], float helmetPrice[])  {          std::cout << "\n\t\t\tPlease choose your helmet size (S/M/L/XL): ";          std::cin >> helmetSize[i];            std::transform(helmetSize[i].begin(), helmetSize[i].end(), helmetSize[i].begin(), ::toupper);          while (std::cin.fail() || (helmetSize[i] != "S" && helmetSize[i] != "M" && helmetSize[i] != "L" && helmetSize[i] != "XL")) {                  std::cin.clear();                  std::cin.ignore(1000, '\n');                  std::cout << "\n\t\t\tPlease choose a valid helmet size (S/M/L/XL): ";                  std::cin >> helmetSize[i];                  std::transform(helmetSize[i].begin(), helmetSize[i].end(), helmetSize[i].begin(), ::toupper);          }            if (helmetSize[i] == "S")                  helmetPrice[i] = 2;          if (helmetSize[i] == "M")                  helmetPrice[i] = 4;          if (helmetSize[i] == "L")                  helmetPrice[i] = 6;          if (helmetSize[i] == "XL")                  helmetPrice[i] = 8;          return helmetPrice[i];  }  float selectSuit(int i, std::string suitSize[], float suitPrice[])  {          std::cout << "\n\t\t\tPlease choose your suit size (S/M/L/XL): ";          std::cin >> suitSize[i];          std::transform(suitSize[i].begin(), suitSize[i].end(), suitSize[i].begin(), ::toupper);          while (std::cin.fail() || (suitSize[i] != "S" && suitSize[i] != "M" && suitSize[i] != "L" && suitSize[i] != "XL")){                  std::cin.clear();                  std::cin.ignore(1000, '\n');                  std::cout << "\n\t\t\tPlease choose a valid suit size (S/M/L/XL): ";                  std::cin >> suitSize[i];                  std::transform(suitSize[i].begin(), suitSize[i].end(), suitSize[i].begin(), ::toupper);          }            if (suitSize[i] == "S")                  suitPrice[i] = 5;          if (suitSize[i] == "M")                  suitPrice[i] = 10;          if (suitSize[i] == "L")                  suitPrice[i] = 15;          if (suitSize[i] == "XL")                  suitPrice[i] = 20;            return suitPrice[i];  }  float selectShoe(int i, int shoeSize[], float shoePrice[])  {          std::cout << "\n\t\t\tPlease choose your shoe size (20 - 35): ";          std::cin >> shoeSize[i];            while (std::cin.fail() || (shoeSize[i] < 20 || shoeSize[i] > 35)){                  std::cin.clear();                  std::cin.ignore(1000, '\n');                  std::cout << "\n\t\t\tPlease choose a valid shoe size (20 - 35): ";                  std::cin >> shoeSize[i];          }            if (shoeSize[i] <= 25)                  shoePrice[i] = 3;          else                  shoePrice[i] = 6;          return shoePrice[i];  }  float calculateTotalGearPrice(int& driverCount, float helmetPrice[], float suitPrice[], float shoePrice[], float gearPrice[])  {          float totalGearPrice = 0;            for (int i = 0; i < driverCount; i++) {                  gearPrice[i] = helmetPrice[i] + suitPrice[i] + shoePrice[i];                  totalGearPrice += gearPrice[i];          }          return totalGearPrice;  }  float setMembershipDiscount()  {          char membership;          std::cout << "\n\t\t\tDo you have a membership? (Y/N): ";          membership = getValidCharacterInput();          if(membership == 'Y')                  return 0.1;          else                  return 0.0;  }  float calculatePrice(int engineCapacity[], int laps, int driverCount, float totalGearPrice, float membershipDiscount)  {          const float pricePerCC = 0.3;          float totalKartPrice = 0;            for (int i = 0; i < driverCount; i++)                  totalKartPrice += (engineCapacity[i] \* pricePerCC) \* laps;            float finalPrice = (totalKartPrice + totalGearPrice) \* (1 - membershipDiscount);          return finalPrice;  } |

1. **SAMPLES OF INPUT AND OUTPUT**

|  |
| --- |
|  |

1. **DISCUSSIONS**
2. **CONLUSION**
3. **PRESENTATION**

The presentation is available on YouTube: [INSERT VIDEO LINK]