| **Name** | Jhaveri Varun Nimitt |
| --- | --- |
| **UID no.** | 2023800042 |
| **Experiment No.** | 8 |

| **AIM:** | Using structures and unions to solve a given problem. |
| --- | --- |
| **Program 1** | |
| **PROBLEM STATEMENT :** | As a team manager, you want to pick the best batsman from your IPL team for an upcoming match. Your system collects data of each player as his name, country and batting average. Then it sorts the list based on batting average and produces the output to you, so you can select the best batsman. [Take input of 11 players] |
| **ALGORITHM:** | **START**  1.1 Define a structure named player with fields:  - name (string)  - country (string)  - battingavg (double)  2.1 Function selectionSort(arr: array of player, n: integer)  2.1.1 For i from 0 to n-1  2.1.1.1 Set max\_idx to i  2.1.1.2 For j from i+1 to n  2.1.1.2.1 If arr[j].battingavg > arr[max\_idx].battingavg  Set max\_idx to j  2.1.1.3 Swap arr[max\_idx] and arr[i]  3.1 Declare an array of player structures named allmyplayers with size 11  3.2 For i from 0 to 11;i < 11; i++  3.2.1 Prompt the user to input player data:  - Enter name  - Enter country  - Enter batting average  - Store input in allmyplayers[i].name, allmyplayers[i].country, allmyplayers[i].battingavg  3.3 Call selectionSort(allmyplayers, 11) to sort the players based on batting averages  4.1 For i from 0 to 11;i < 11; i++  4.1.1 Print allmyplayers[i].name, allmyplayers[i].country, allmyplayers[i].battingavg  5.1 Print "The best batsman from your IPL team for upcoming match is: "  5.2 Print allmyplayers[0].name  **END** |
| **PROGRAM:** | #include <stdio.h> #include <string.h> #include <stdlib.h>  struct player {  char name[20];  char country[20];  double battingavg; };   void selectionSort(struct player arr[], int n) {  int i, j, max\_idx;  struct player temp;  for (i = 0; i < n-1; i++)  {  max\_idx = i;  for (j = i+1; j < n; j++)  {  if (arr[j].battingavg > arr[max\_idx].battingavg)  {  max\_idx = j;  }  }  temp = arr[max\_idx];  arr[max\_idx] = arr[i];  arr[i] = temp;  } }  int main() {  struct player allmyplayers[11];  printf("Enter name, country and batting average for all 11 players (separated by a space): \n");  for (int i = 0; i < 11; i++)  {  scanf("%s",allmyplayers[i].name);  scanf("%s",allmyplayers[i].country);  scanf("%lf", &allmyplayers[i].battingavg);  }  selectionSort(allmyplayers, 11);  printf("\nSorted players by batting average:\n");  for (int i = 0; i < 11; i++)  {  printf("%s (%s): %.2lf", allmyplayers[i].name, allmyplayers[i].country, allmyplayers[i].battingavg);  printf("\n");  }    printf("The best batsman from your IPL team for upcoming match is: %s", allmyplayers[0].name);  return 0; } */\* Harry AUS 34.56 Hitesh IND 78.90 Kanye WI 12.34 Hardik IND 56.78 Virat IND 90.12 Babar PAK 45.67 Sachin IND 89.01 Kautubh JAP 23.45 Kshitij MLM 67.89 Marshall NZ 45.67 Sindhu IND 78.90 \*/* |
| **RESULT:** | |
| **Program 2** | |
| **PROBLEM STATEMENT :** | An airline reservation system maintains records for possible flights consisting of:  STARTING POINT 3 character code  DESTINATION 3 character code  STARTING TIME integer on scale 0001 – 2400  ARRIVAL TIME integer on scale 0001 – 2400  SEATS positive integer in suitable range.  Your program is to read 20 such records followed by queries of the form STARTING  POINT– DESTINATION, one to a line. For each query, find whether there is a possible flight with a seat available; if so, reduce the number of seats by one and print out the flight details (or an apology). |
| **ALGORITHM:** | **START**  1.1 Define a structure named Flight with fields:  - og (string): Departure city  - dest (string): Arrival city  - **start**\_time (integer): Departure time  - arrival\_time (integer): Arrival time  - seats (integer): Number of seats in the flight  1.2 Define color constants for formatting output:  - COLOR\_BOLD\_SLOW\_BLINKING\_RED  - COLOR\_BOLD\_SLOW\_BLINKING\_BLUE  - COLOR\_OFF  - COLOR\_BOLD  2.1 Function get\_flight\_details(flights: array of Flight, n: integer)  2.1.1 Display prompt: "Enter details for all 20 flights in the format of Departure city, Arrival city, Departure time, Arrival time, Number of seats in flight:"  2.1.2 For i from 0 to n; i++  2.1.2.1 Read input for flights[i].og, flights[i].dest, flights[i].start\_time, flights[i].arrival\_time, flights[i].seats  2.1.2.2 If flights[i].start\_time == 2400 or flights[i].arrival\_time == 2400, display error and terminate program  3.1 Function book\_flight(flights: array of Flight, n: integer)  3.1.1 While true  3.1.1.1 Display prompt: "Enter your query in the format of | Departure City Arrival City |:"  3.1.1.2 Read input for wherego and whereland  3.1.1.3 For j from 0 to n ; j++  3.1.1.3.1 If strcmp(flights[j].og == wherego and flights[j].dest == whereland and flights[j].seats > 0) ==0  3.1.1.3.1.1 Decrement flights[j].seats  3.1.1.3.1.2 Display booking confirmation with formatted information  3.1.1.3.1.3 Break from the loop  3.1.1.4 If j == n, display "No flights available from wherego to whereland"  3.1.1.5 Display prompt: "Press C to end, otherwise press D to start another query:"  3.1.1.6 Read input for exitcheck  3.1.1.7 If exitcheck == 'C' or 'c', break from the loop  4.1 Function main()  4.1.1 Declare an array of Flight structures named flights with a size of 20  4.1.2 Set n to 20  4.1.3 Call get\_flight\_details(flights, n) to input flight details  4.1.4 Call book\_flight(flights, n) to handle flight bookings  **END** |
| **PROGRAM:** | #include <stdio.h> #include <string.h> #include <stdlib.h> #define COLOR\_BOLD\_SLOW\_BLINKING\_RED "\e[1;5;31m" #define COLOR\_BOLD\_SLOW\_BLINKING\_BLUE "\e[1;5;34m" #define COLOR\_OFF "\e[m" #define COLOR\_BOLD "\e[1m"  struct Flight {  char og[20];  char dest[20];  int start\_time;  int arrival\_time;  int seats; };  void get\_flight\_details(struct Flight flights[], int n) {  int i;  printf("Enter details for all 20 flights in the format of Departure city, Arrival city, Departure time, Arrival time, Number of seats in flight: \n");  for (i = 0; i < n; i++)  {    scanf("%3s %3s %d %d %d", flights[i].og, flights[i].dest, &flights[i].start\_time, &flights[i].arrival\_time, &flights[i].seats);  if(flights[i].start\_time == 2400 || flights[i].arrival\_time==2400)  {  printf("Error: Time cannot be above 2400 or 2400, terminating program.");  exit(0);  }  } }  void book\_flight(struct Flight flights[], int n) {  int j;  char wherego[20], whereland[20];  char exitcheck;   while (1)  {  printf("\nEnter your query in the format of | Departure City Arrival City |:");  scanf("%3s %3s", wherego, whereland);  for (j = 0; j < n; j++)  {  if (strcmp(flights[j].og, wherego) == 0 && strcmp(flights[j].dest, whereland) == 0 && flights[j].seats > 0)  {  flights[j].seats--;  printf(COLOR\_BOLD"Booking Confirmed:\n\n"COLOR\_OFF);  printf("Your flight departs from");  printf(COLOR\_BOLD\_SLOW\_BLINKING\_RED" %s ",flights[j].og );  printf(COLOR\_OFF "at");  printf(COLOR\_BOLD\_SLOW\_BLINKING\_BLUE" %d ",flights[j].start\_time);  printf(COLOR\_OFF"and arrives at");  printf(COLOR\_BOLD\_SLOW\_BLINKING\_RED" %s ",flights[j].dest);  printf(COLOR\_OFF "at");  printf(COLOR\_BOLD\_SLOW\_BLINKING\_BLUE" %d.",flights[j].arrival\_time);  printf(COLOR\_OFF "\n\nThe amount of seats left in the same flight is: %d\n", flights[j].seats);  break;  }  }  if (j == n)  {  printf("\nNo flights available from %s to %s\n", wherego, whereland);  }   printf("\nPress C to end, otherwise press D to start another query: ");  scanf(" %c", &exitcheck);  if (exitcheck == 'C' || exitcheck == 'c')  {  break;  }  else  {  continue;  }  } }  int main() {  struct Flight flights[20];  int n = 20;    get\_flight\_details(flights, n);  book\_flight(flights, n);   return 0; } |
| **RESULT:**  [**Better resolution**](https://photos.app.goo.gl/E9BExzdm1t2F1DyV8)  **(show the blinking text) to showcase the cool stuff I found at :** [**bash commands**](https://github.com/ElectricRCAircraftGuy/eRCaGuy_dotfiles/blob/master/useful_scripts/git-diffn.sh#L126-L138) | |
| **CONCLUSION:** | **I have understood the difference between structures and unions and learned how to use them to solve a problem efficiently.** |