Introduction to Employee Shift Planner

An Employee Shift Planner is a tool or system designed to organize and manage work schedules for employees across different shifts. It helps businesses ensure that all roles are covered efficiently while balancing employee availability, labor laws, and company needs.

With a shift planner, managers can easily assign shifts, track attendance, manage time-off requests, and adjust schedules based on workload demands. Modern shift planners often include digital features like automated scheduling, real-time updates, mobile access, and notifications, making the process smoother for both employers and employees.

Effective shift planning enhances productivity, reduces scheduling conflicts, boosts employee satisfaction, and helps optimize operational costs.

Algorithm:

1. Setup:

List all employees and their availability for each shift and day.

2. Make Everyone Available:

Set every employee as available for all shifts at first.

(Example: Bob is made unavailable for night shifts.)

3. Assign Shifts:

For each day:

For each shift (Morning, Evening, Night):

Find the first available employee who isn't already assigned that day.

Assign that employee to the shift.

4. Show the Schedule:

For each day:

Print who is working which shift.

5. Run the Program:

Create employees.

Set availability.

Assign shifts.

Print the weekly plan.

Source code:

#include <stdio.h>

#include <string.h>

#define MAX\_EMPLOYEES 10

#define DAYS 7

#define SHIFTS 3

const char\* shift\_names[SHIFTS] = {"Morning", "Evening", "Night"};

const char\* day\_names[DAYS] = {"Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday"};

typedef struct {

char name[50];

int availability[DAYS][SHIFTS]; // 1 if available, 0 otherwise

} Employee;

void initialize\_availability(Employee employees[], int num\_employees) {

for (int i = 0; i < num\_employees; i++) {

for (int d = 0; d < DAYS; d++) {

for (int s = 0; s < SHIFTS; s++) {

employees[i].availability[d][s] = 1; // All available by default

}

}

}

// Example: Making Bob unavailable for Night shifts

for (int d = 0; d < DAYS; d++) {

employees[1].availability[d][2] = 0;

}

}

void assign\_shifts(Employee employees[], int num\_employees, int schedule[DAYS][SHIFTS]) {

int assigned[DAYS][MAX\_EMPLOYEES] = {0};

for (int d = 0; d < DAYS; d++) {

for (int s = 0; s < SHIFTS; s++) {

for (int e = 0; e < num\_employees; e++) {

if (employees[e].availability[d][s] && !assigned[d][e]) {

schedule[d][s] = e;

assigned[d][e] = 1;

break;

}

}

}

}

}

void print\_schedule(Employee employees[], int schedule[DAYS][SHIFTS]) {

printf("Shift Schedule:\n");

for (int d = 0; d < DAYS; d++) {

printf("%s:\n", day\_names[d]);

for (int s = 0; s < SHIFTS; s++) {

int e = schedule[d][s];

printf(" %s: %s\n", shift\_names[s], employees[e].name);

}

printf("\n");

}

}

int main() {

Employee employees[MAX\_EMPLOYEES] = {

{"Alice"},

{"Bob"},

{"Carol"},

{"Dave"},

{"Eve"}

};

int num\_employees = 5;

int schedule[DAYS][SHIFTS];

initialize\_availability(employees, num\_employees);

assign\_shifts(employees, num\_employees, schedule);

print\_schedule(employees, schedule);

return 0;

}

Output :

Shift Schedule:

Monday:

Morning: Alice

Evening: Bob

Night: Carol

Tuesday:

Morning: Dave

Evening: Eve

Night: Alice

Wednesday:

Morning: Bob

Evening: Carol

Night: Dave

Thursday:

Morning: Eve

Evening: Alice

Night: Bob

Friday:

Morning: Carol

Evening: Dave

Night: Eve

Saturday:

Morning: Alice

Evening: Bob

Night: Carol

Sunday:

Morning: Dave

Evening: Eve

Night: Alice

Conclusion:

The Employee Shift Planner program successfully creates a basic weekly schedule by assigning available employees to morning, evening, and night shifts. It ensures that no employee is double-booked on the same day and respects individual availability constraints, such as Bob not working night shifts. This system helps automate shift planning, saves time, and reduces errors compared to manual scheduling. With further improvements, such as more advanced assignment rules or user inputs, this planner can be adapted for real-world business needs.