EAST WEST UNIVERSITY

Department of CSE

Algorithm- **Fall 2023**

Project Title: **Balancing Workload Equitably**

Submitted by:

Name: Amin Talukder
ID:2019-3-60-014

Name: Sayemuzzaman Siam
ID:2019-3-60-011

Name: Sadi Mohammad Kabbo
ID:2018-3-60-007

Name: Sajeeb Karmoker
ID:2019-1-60-095

Submitted to:

Amit Mandal

Lecturer

East West University

Project Title: Balancing Workload Equitably

Scenario:

At Tech Solutions, CEO Sam noticed an uneven workload among teams affecting project timelines. To address this, the HR team developed an algorithm to resolve the issue. They gathered employee profiles, including names and roles, along with task details such as user assignments and processing times.

The algorithm was designed to sort tasks based on users and processing times, ensuring fairness in workload distribution and also calculate the fairness score.

Algorithm:

```
#include <stdio.h>
#include <string.h>
#define MAX_USERS 50
#define MAX_TASKS 100

struct Task {
    char user[50];
    int processing_time;
};
```

```
void swap(struct Task *a, struct Task *b) {
    struct Task temp = *a;
    *a = *b;
    *b = temp;
}

void sortTasks(struct Task tasks[], int num_tasks) {
    for (int i = 0; i < num_tasks - 1; i++) {
        for (int j = 0; j < num_tasks - i - 1; j++) {
            if (tasks[j].processing_time > tasks[j + 1].processing_time ||
                (tasks[j].processing_time == tasks[j + 1].processing_time &&
                 strcmp(tasks[j].user, tasks[j + 1].user) > 0)) {
                swap(&tasks[j], &tasks[j + 1]);
            }
        }
    }
}

int calculateFairnessScore(int user_times[], int num_users, struct Task
tasks[], int num_tasks) {
    int dp[MAX_USERS + 1][MAX_TASKS + 1];

    for (int i = 0; i <= num_users; i++) {
        for (int j = 0; j <= num_tasks; j++) {
            if (i == 0 || j == 0)
```

```
            dp[i][j] = 0;
        else if (user_times[i - 1] == tasks[j - 1].processing_time)
            dp[i][j] = dp[i - 1][j - 1] + 1;
        else
            dp[i][j] = (dp[i - 1][j] > dp[i][j - 1]) ? dp[i - 1][j] : dp[i][j - 1];
      }
   }

   return dp[num_users][num_tasks];
}

void optimizeSchedule(struct Task users[], struct Task tasks[], int num_users,
int num_tasks) {
   sortTasks(tasks, num_tasks);

   int user_times[MAX_USERS];
   memset(user_times, 0, sizeof(user_times));

   for (int i = 0; i < num_tasks; i++) {
      for (int j = 0; j < num_users; j++) {
         if (strcmp(tasks[i].user, users[j].user) == 0) {
            user_times[j] += tasks[i].processing_time;
            break;
         }
      }
```

```
    }

    int fairness_score = calculateFairnessScore(user_times, num_users, tasks,
num_tasks);

    printf("\nOptimized Schedule:\n");
    for (int i = 0; i < num_tasks; i++) {
        printf("User: %s, Processing Time: %d\n", tasks[i].user,
tasks[i].processing_time);
    }
    printf("\nFairness Score: %d\n", fairness_score);
}

int main() {
    int num_users, num_tasks;

    printf("Enter the number of users: ");
    scanf("%d", &num_users);

    struct Task users[MAX_USERS];
    for (int i = 0; i < num_users; i++) {
        printf("Enter user %d: ", i + 1);
        scanf("%s", users[i].user);
    }
```

```
    printf("Enter the number of tasks: ");
    scanf("%d", &num_tasks);

    struct Task tasks[MAX_TASKS];
    for (int i = 0; i < num_tasks; i++) {
        printf("Enter user for task %d: ", i + 1);
        scanf("%s", tasks[i].user);
        printf("Enter processing time for task %d: ", i + 1);
        scanf("%d", &tasks[i].processing_time);
    }

    optimizeSchedule(users, tasks, num_users, num_tasks);

    return 0;
}
```

Conclusion:

The adoption of this task allocation system has proven to be instrumental in achieving fairer task distribution among employees. It has not only enhanced productivity but has also fostered a positive work environment, aligning with the company's commitment to equitable practices.