DAA – Practical 4

Name: Aditya.Rajesh.Wanwade

Roll no :27 && Batch:C2

Aim : a) Problem: Consider a scenario where a student has limited time to study for exams and wants to maximize the total score achieved in a set amount of time. Each topic represents an item with a certain study time and corresponding value(score). The Fractional Knapsack Problem can be applied to help the student strategically allocate their study time to maximize the overall score within the available time frame. Implement the solution for this problem.

Code of a) problem :

```c
#include <stdio.h>
#include <stdlib.h>

struct array
{
    int score;
    int time;
    int id;
};

int cmp(const void *a, const void *b)
{
    double r1 = ((double)((struct array *)a)->score) / ((struct array *)a)->time;
    double r2 = ((double)((struct array *)b)->score) / ((struct array *)b)->time;
    return (r1 < r2) - (r1 > r2);
}

double maxScore(struct array s[], int maxtime, int size)
{
    qsort(s, size, sizeof(struct array), cmp);

    int cur_weight = 0;
    double final_val = 0.0;
    for (int i = 0; i < size; i++)
    {
```

```
        if (cur_weight + s[i].time <= maxtime)
        {
            cur_weight += s[i].time;
            final_val += s[i].score;
            printf("%d\t",s[i].id);
        }
        else
        {
            int remain = maxtime - cur_weight;
            final_val += s[i].score * ((double)remain / s[i].time);
            printf("%d\t",s[i].id);
            break;
        }
    }
    return final_val;
}

int main()
{
    int i = 0;
    int maxtime, size;

    printf("Enter the number of subjects: ");
    scanf("%d", &size);
    struct array s[size];

    printf("Enter the score of each topic: ");
    for (i = 0; i < size; i++)
    {
        scanf("%d", &(s[i].score));
    }

    printf("Enter the time of each topic: ");
    for (i = 0; i < size; i++)
    {
        scanf("%d", &(s[i].time));
    }
     for (i = 0; i < size; i++)
    {
        s[i].id=i+1;
    }


    printf("Enter the maximum time: ");
    scanf("%d", &maxtime);
printf("The selected topics are:");
    double maximumScore = maxScore(s, maxtime, size);
    printf("\nThe maximum score is:%f", maximumScore);
```

```
    return 0;
}
```

Output:

```
PS E:\DAA> cd "e:\DAA\" ; if ($?) { gcc knapsack.c -o knapsack } ;
 if ($?) { .\knapsack }
Enter the number of subjects: 7
Enter the score of each topic: 10
5
15
7
6
18
3
Enter the time of each topic: 2
3
5
7
1
4
1
Enter the maximum time: 15
The selected topics are:5       1       6       7       3       2
The maximum score is:55.333333
PS E:\DAA>
```

b) Problem: Consider a scenario where a conference organizer wants to schedule multiple talks within a limited time frame, with each talk having a start and end time. The organizer aims to maximize the number of talks without overlapping. The greedy approach can be applied to efficiently select the talks and create an optimal schedule. Implement the solution for this problem.

Code of b) problem:

```c
#include <stdio.h>
#include <stdlib.h>

struct Talk {
    int startTime;
    int endTime;
    int id;
};

int compareTalks(const void *a, const void *b) {
    return ((struct Talk *)a)->endTime - ((struct Talk *)b)->endTime;
}
```

```c
void scheduleTalks(struct Talk talks[], int n) {

    qsort(talks, n, sizeof(struct Talk), compareTalks);
    int i = 0;
    printf("Selected talks:\n");
    printf("Talk %d: Start Time = %d, End Time = %d\n",talks[i].id,
talks[i].startTime, talks[i].endTime);
    for (int j = 1; j < n; j++) {

        if (talks[j].startTime >= talks[i].endTime) {
            printf("Talk %d: Start Time = %d, End Time = %d\n",talks[j].id,
talks[j].startTime, talks[j].endTime);
            i = j;
        }
    }
}

int main() {
    int n;
    printf("Enter the number of talks: ");
    scanf("%d", &n);

    struct Talk talks[n];


    for (int i = 0; i < n; i++) {
        printf("Enter start time and end time for Talk %d: ", i + 1);
        talks[i].id=i+1;
        scanf("%d %d", &talks[i].startTime, &talks[i].endTime);
    }
    scheduleTalks(talks, n);

    return 0;
}
```

Output:

```
PS E:\DAA> cd "e:\DAA\" ; if ($?) { gcc activitySelection.c -o activitySelection } ; if ($?) { .\activit
ySelection }
Enter the number of talks: 6
Enter start time and end time for Talk 1: 0 6
Enter start time and end time for Talk 2: 3 4
Enter start time and end time for Talk 3: 1 2
Enter start time and end time for Talk 4: 5 9
Enter start time and end time for Talk 5: 5 7
Enter start time and end time for Talk 6: 8 9
Selected talks:
Talk 3: Start Time = 1, End Time = 2
Talk 2: Start Time = 3, End Time = 4
Talk 5: Start Time = 5, End Time = 7
Talk 6: Start Time = 8, End Time = 9
```