

## Experiment No.3

Title:-Knapsack Problem using Greedy Method.

**Problem:-** Allocation in Cloud Computing Assigning limited server resources (CPU, memory, bandwidth) to tasks to maximize efficiency or profit.

Programm:-

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

```
typedef struct {
```

```
    char name[20];
```

```
    float value;    // profit or importance
```

```
    float resource; // resource required (CPU units)
```

```
    float ratio;    // value per resource
```

```
} Task;
```

```
// Function to sort tasks by value/resource ratio (descending)
```

```
void sortTasks(Task tasks[], int n) {
```

```
    for(int i = 0; i < n - 1; i++) {
```

```
        for(int j = i + 1; j < n; j++) {
```

```
            if(tasks[j].ratio > tasks[i].ratio) {
```

```
                Task temp = tasks[i];
```

```
                tasks[i] = tasks[j];
```

```
                tasks[j] = temp;
```

```
            }
```

```
        }
```

```
    }
```

```
}
```

```
// Greedy fractional allocation
```

```

void allocateResources(Task tasks[], int n, float totalResource) {

    float totalValue = 0;

    printf("Task Allocation:\n");

    for(int i = 0; i < n; i++) {

        if(totalResource == 0) break;

        if(tasks[i].resource <= totalResource) {

            // Assign full task

            totalResource -= tasks[i].resource;

            totalValue += tasks[i].value;

            printf("%s -> Fully Allocated\n", tasks[i].name);

        } else {

            // Assign fractional task

            float fraction = totalResource / tasks[i].resource;

            totalValue += tasks[i].value * fraction;

            printf("%s -> Partially Allocated: %.2f%%\n", tasks[i].name, fraction*100);

            totalResource = 0;

        }

    }

    printf("\nTotal Value Achieved: %.2f\n", totalValue);

}

int main() {

    Task tasks[] = {

        {"Task1", 60, 10, 0},

        {"Task2", 100, 20, 0},

        {"Task3", 120, 30, 0}

    };

```

```
int n = sizeof(tasks) / sizeof(tasks[0]);

float totalResource = 50; // total CPU units available

// Calculate value/resource ratio

for(int i = 0; i < n; i++) {

    tasks[i].ratio = tasks[i].value / tasks[i].resource;

}

// Sort tasks by ratio

sortTasks(tasks, n);

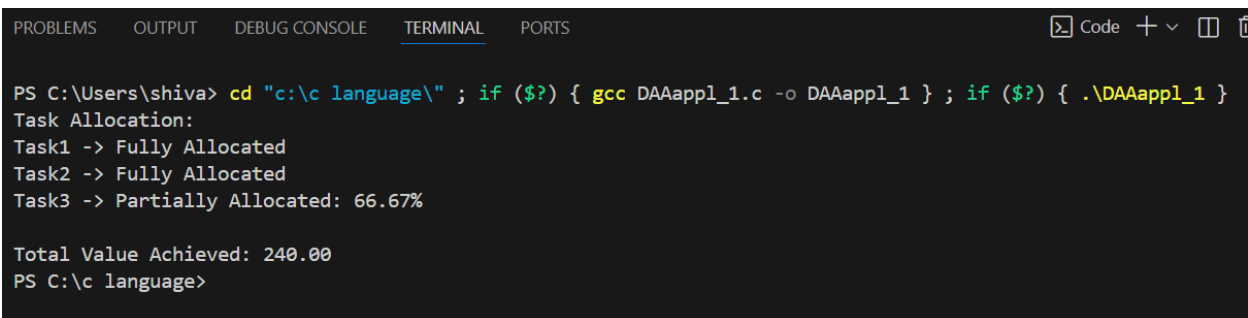
// Allocate resources greedily

allocateResources(tasks, n, totalResource);

return 0;

}
```

Output:-



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  Code + - []
```

```
PS C:\Users\shiva> cd "c:\c language\" ; if ($?) { gcc DAAappl_1.c -o DAAappl_1 } ; if ($?) { .\DAAappl_1 }
Task Allocation:
Task1 -> Fully Allocated
Task2 -> Fully Allocated
Task3 -> Partially Allocated: 66.67%

Total Value Achieved: 240.00
PS C:\c language>
```