Name: Ashmit Thawait
Roll No: 102203790
Group: 2CO17

Assignment 2 – Greedy Approach

Q1. Activity Selection using Greedy Approach:

```
#include <bits/stdc++.h> using
namespace std;
 void printMaxActivities(int s[], int f[], int n)
{ int i, j; cout << "Selected activities: " <<
endl;

    i = 0;
    cout << i << " ";

    for (j = 1; j < n; j++) {
        if (s[j] >= f[i]) {
        cout << j << " "; i =
        j;
        }
    } } int main() { int s[] = {
1, 2, 3, 4, 5, 6 }; int f[] = { 3,
4, 5, 6, 7, 8 };
    int n = sizeof(s) / sizeof(s[0]);

    printMaxActivities(s, f, n);
    return 0;
}
```

Output:

Selected activities: 0 2 4

Q2. Job Sequencing using Greedy Approach

```
#include <bits/stdc++.h>
using namespace std;
 struct Job{char id;int deadline;int profit;}; bool
comparison(Job a, Job b){return (a.profit > b.profit);}

void jobSequencing(Job arr[], int n){ sort(arr, arr + n, comparison); int
    maxDeadline = 0; for (int i = 0; i < n; i++) {maxDeadline = max(maxDeadline,
    arr[i].deadline);} char result[maxDeadline];bool slot[maxDeadline]; for (int
    i = 0; i < maxDeadline; i++) {slot[i] = false;} for (int i = 0; i < n; i++) {
    for (int j = min(maxDeadline, arr[i].deadline) - 1; j >= 0; j--) { if
    (!slot[j]) { result[j] = arr[i].id; slot[j] = true; break;
            }
        } } cout << "Job sequence for maximum
    profit: "; for (int i = 0; i < maxDeadline;
    i++) { if (slot[i]) { cout << result[i] <<
    " ";
        }
    }
}
int main(){
    Job arr[] = {{'a', 2, 100}, {'b', 1, 19}, {'c', 2, 27},
                {'d', 1, 25}, {'e', 3, 15}};
    int n = sizeof(arr) /
    sizeof(arr[0]); jobSequencing(arr,
    n); return 0;
}
```

Output :

Job sequence for maximum profit: c a e

Q3. Fractional Knapsack

```cpp
#include <bits/stdc++.h> using
namespace std;
 struct  Item  {
int  value;  int
weight;
};
bool cmp(Item a, Item b) { double ratio1 =
    (double)a.value / a.weight; double ratio2 =
    (double)b.value / b.weight; return ratio1 >
    ratio2;
}
double fractionalKnapsack(Item arr[], int n, int capacity) {
    sort(arr, arr + n, cmp); double totalValue = 0.0; int
    currentWeight = 0; for (int i = 0; i < n; i++) { if
    (currentWeight + arr[i].weight <= capacity) {
    currentWeight += arr[i].weight; totalValue +=
    arr[i].value;
        } else { int remainingWeight = capacity -
            currentWeight;
            totalValue += arr[i].value * ((double)remainingWeight / arr[i].weight);
            break;
        } }
    return totalValue;
}
int main() { int
    capacity = 50;
    Item arr[] = {{60, 10}, {100, 20}, {120, 30}};
    int n = sizeof(arr) / sizeof(arr[0]);
    cout << "Maximum value in Knapsack = " << fractionalKnapsack(arr, n, capacity) << endl;
    return 0;
}
```

Output :

Maximum value in Knapsack = 240 Q4. Huffman Coding

```
#include <bits/stdc++.h> using
namespace std;
 struct MinHeapNode
{ char data;
unsigned freq;
    MinHeapNode *left, *right;
};
struct compare { bool operator()(MinHeapNode* l,
    MinHeapNode* r)
{ return (l->freq > r->freq); }
};
void printCodes(MinHeapNode* root, string str)
    { if (!root) return;
    if (root->data != '$') cout << root->data <<
        ": " << str << "\n";
    printCodes(root->left, str + "0");
    printCodes(root->right, str + "1");
}
void HuffmanCodes(char data[], int freq[], int
size) {
    MinHeapNode *left, *right, *top;
```

```
    priority_queue<MinHeapNode*,
vector<MinHeapNode*>, compare> minHeap;
 for   (int   i   =   0;   i   <   size;   ++i)
    minHeap.push(new MinHeapNode{data[i],
static_cast<unsigned>(freq[i]), nullptr, nullptr});

    while (minHeap.size() != 1) {
        left   =   minHeap.top();
        minHeap.pop();   right   =
        minHeap.top();
        minHeap.pop();
        top = new MinHeapNode{'$', left->freq +
right->freq, left, right};
        minHeap.push(top);
    }
    printCodes(minHeap.top(), "");
}
int main() { char data[] = {'a', 'b', 'c', 'd',
    'e', 'f'}; int freq[] = {5, 9, 12, 13, 16,
    45}; int size = sizeof(data) /
    sizeof(data[0]); HuffmanCodes(data, freq,
    size); return 0;
}
```

Output:

f: 0 c: 100 d: 101

a: 1100 b: 1101

e: 111