**Algorithms**

**Laboratory Task-4**

**Submission Deadline** – As announced in the class

**Submission Guidelines**-

* Rename the file to your id only. If your id is 18-XXXXX-1, then the file name must be 18-XXXXX-1.docx.
* Must submit within the given deadline in VUES to the section named Lab Tak-2
* Must include resources for all the section named ‘Code’ and ‘Output (screenshot)’ in the table.

|  |
| --- |
| Question-1 – **Implement Coin Changing Problem with Greedy Approach** |
| **Pseudocode**   1. Sort the array of coins in decreasing order. 2. Initialize result as empty. 3. Find the largest denomination that is smaller than current amount. 4. Add found denomination to result. Subtract value of found denomination from amount. 5. If amount becomes 0, then print result. 6. Else repeat steps 3 and 4 for new value of V. |
| **Code**  **#include <bits/stdc++.h>**  **using namespace std;**  **// All denominations of Indian Currency**  **int deno[] = { 1, 2, 5, 10, 20,**  **50, 100, 500, 1000 };**  **int n = sizeof(deno) / sizeof(deno[0]);**  **void findMin(int V)**  **{**  **sort(deno, deno + n);**  **// Initialize result**  **vector<int> ans;**  **// Traverse through all denomination**  **for (int i = n - 1; i >= 0; i--) {**  **// Find denominations**  **while (V >= deno[i]) {**  **V -= deno[i];**  **ans.push\_back(deno[i]);**  **}**  **}**  **// Print result**  **for (int i = 0; i < ans.size(); i++)**  **cout << ans[i] << " ";**  **}**  **// Driver program**  **int main()**  **{**  **int n = 93;**  **cout << "Following is minimal"**  **<< " number of change for " << n**  **<< ": ";**  **findMin(n);**  **return 0;**  **}** |
| **Output (Screenshot)** |

|  |
| --- |
| Question-2 – **Implement Activity Selection Problem with Greedy Approach** |
| **Pseudocode**  Set i = 0; //pointing at first element  for j = 1 to n-1 do  if start time of j >= finish time of i then  Print j  Set i = j  endif  endfor |
| **Code#include<iostream>**  **using namespace std;**  **struct Activity**  **{**  **int startTime, finishTime;**  **};**  **void SWAP(struct Activity \*p, struct Activity \*q)**  **{**  **struct Activity t;**  **t=\*p;**  **\*p=\*q;**  **\*q=t;**  **}**  **void bubbleSort(struct Activity a[], int n)**  **{**  **int pass,i;**  **for(pass=1;pass<n;pass++)**  **{**  **for(i=0;i<n;i++)**  **{**  **if(a[i].finishTime>a[i+1].finishTime)**  **{**  **SWAP(&a[i],&a[i+1]);**  **}**  **}**  **}**  **}**  **void activitySelection(struct Activity a[], int n)**  **{**  **int i,j;**  **i=0;**  **cout<<a[0].startTime<<" "<<a[0].finishTime<<endl;**  **for(j=i+1;j<n;j++)**  **{**  **if(a[j].startTime>=a[i].finishTime)**  **{**  **cout<<a[j].startTime<<" "<<a[j].finishTime<<endl;**  **i=j;**  **}**  **}**  **}**  **int main()**  **{**  **int n,i;**  **cout<<"Enter Number of Activities: ";**  **cin>>n;**  **struct Activity act[n];**  **for(i=0;i<n;i++)**  **{**  **cout<<"Enter start time: ";**  **cin>>act[i].startTime;**  **cout<<"Enter finish time: ";**  **cin>>act[i].finishTime;**  **}**  **bubbleSort(act,n);**  **/\*cout<<"After sorting"<<endl;**  **for(i=0;i<n;i++)**  **{**  **cout<<act[i].startTime<<" "<<act[i].finishTime<<endl;**  **}\*/**  **cout<<"Selected Activities: "<<endl;**  **activitySelection(act,n);**  **}** |
| **Output (Screenshot)** |

|  |
| --- |
| Question-3 – **Implement Fractional Knapsack Problem with Greedy Approach** |
| **Pseudocode**  Fractional knapsack class 13 |
| **Code**  **#include <iostream>**  **#include<conio.h>**  **using namespace std;**  **int getMaxValue(int x, int y)**  **{**  **if(x>y)**  **{**  **return x;**  **}**  **else**  **return y;**  **}**  **void knapsack(int v[], int wt[], int n, int M)**  **{**  **int K[n+1][M+1];**  **int i,w;**  **for(w=0;w<=M;w++)**  **{**  **K[0][w]=0;**  **}**  **for(i=0;i<=n;i++)**  **{**  **K[i][0]=0;**  **}**  **for(i=1;i<=n;i++)**  **{**  **for(w=1;w<=M;w++)**  **{**  **if(wt[i]<=w)**  **{**  **K[i][w] = getMaxValue(K[i-1][w], v[i] + K[i-1][w - wt[i]]);**  **}**  **else**  **K[i][w]= K[i-1][w];**  **}**  **}**  **int p,q;**  **for(p=0;p<=n;p++)**  **{**  **for(q=0;q<=M;q++)**  **{**  **cout<<K[p][q]<<"\t";**  **}**  **cout<<endl;**  **}**  **cout<<endl;**  **cout<<"Maximum profit: "<<K[n][M]<<endl;**  **}**  **int main()**  **{**  **int n,i,M;**  **cout<<"Enter number of items: ";**  **cin>>n;**  **cout<<endl;**  **int wt[n+1], v[n+1];**  **for(i=1;i<=n;i++)**  **{**  **cout<<"Enter weight for item "<<i<<": ";**  **cin>>wt[i];**  **cout<<"Enter value for item "<<i<<": ";**  **cin>>v[i];**  **}**  **cout<<endl;**  **cout<<"Enter Knapsack Capacity: ";**  **cin>>M;**  **knapsack(v,wt,n,M);**  **getch();**  **return 0;**  **}** |
| **Output (Screenshot)** |