Programs for DAA lab

Program 10

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

#include<conio.h>

struct hash \*hashTable = NULL;

int eleCount = 0;

struct node {

int key;

struct node \*next;

};

struct hash {

struct node \*head;

int count;

};

struct node \* createNode(int key)

{

struct node \*newnode;

newnode = (struct node \*)malloc(sizeof(struct node));

newnode->key = key;

newnode->next = NULL;

return newnode;

}

void insertToHash(int key)

{

int hashIndex = key % eleCount;

struct node \*newnode = createNode(key);

if (!hashTable[hashIndex].head)

{

hashTable[hashIndex].head = newnode;

hashTable[hashIndex].count = 1;

return;

}

newnode->next = (hashTable[hashIndex].head);

hashTable[hashIndex].head = newnode;

hashTable[hashIndex].count++;

return;

}

void deleteFromHash(int key)

{

int hashIndex = key % eleCount, flag = 0;

struct node \*temp, \*myNode;

myNode = hashTable[hashIndex].head;

if (!myNode)

{

printf("Given data is not present in hash Table!!\n");

return;

}

temp = myNode;

while (myNode != NULL)

{

if (myNode->key == key)

{

flag = 1;

if (myNode == hashTable[hashIndex].head)

hashTable[hashIndex].head = myNode->next;

else

temp->next = myNode->next;

hashTable[hashIndex].count--;

free(myNode);

break;

}

temp = myNode;

myNode = myNode->next;

}

if (flag)

printf("Data deleted successfully from Hash Table\n");

else

printf("Given data is not present in hash Table!!!!\n");

return;

}

void searchInHash(int key)

{

int hashIndex = key % eleCount, flag = 0;

struct node \*myNode;

myNode = hashTable[hashIndex].head;

if (!myNode)

{

printf("Search element unavailable in hash table\n");

return;

}

while (myNode != NULL)

{

if (myNode->key == key)

{

printf("KEY FOUND : %d\n", myNode->key);

flag = 1;

break;

}

myNode = myNode->next;

}

if (!flag)

printf("Search element unavailable in hash table\n");

return;

}

void display()

{

struct node \*myNode;

int i;

for (i = 0; i < eleCount; i++)

{

if (hashTable[i].count == 0)

continue;

myNode = hashTable[i].head;

if (!myNode)

continue;

printf("\nData at index %d in Hash Table:\n", i);

while (myNode != NULL)

{

printf("%-12d", myNode->key);

myNode = myNode->next;

}

}

return;

}

main()

{

int n,key,ch;

printf("Enter the number of elements:");

scanf("%d", &n);

eleCount = n;

hashTable = (struct hash \*)calloc(n, sizeof (struct hash));

while (1)

{

printf("\n1. Insertion\t2. Deletion\n");

printf("3. Searching\t4. Display\n5. Exit\n");

printf("Enter your choice:");

scanf("%d", &ch);

switch (ch)

{

case 1:

printf("Enter the key value:");

scanf("%d", &key);

getchar();

insertToHash(key);

break;

case 2:

printf("Enter the key to perform deletion:");

scanf("%d", &key);

deleteFromHash(key);

break;

case 3:

printf("Enter the key to search:");

scanf("%d", &key);

searchInHash(key);

break;

case 4:

display();

break;

case 5:

exit(0);

default:

printf("wrong option!!\n");

break;

}

}

getch();

}

Program 11(A)

#include<stdio.h>

#include<conio.h>

int max(int a, int b)

{

return (a > b) ? a : b;

}

int knapSack(int W, int wt[], int val[], int n)

{

if (n == 0 || W == 0)

return 0;

if (wt[n - 1] > W)

return knapSack(W, wt, val, n - 1);

else

return max(

val[n - 1] + knapSack(W - wt[n - 1], wt, val, n - 1),

knapSack(W, wt, val, n - 1));

}

void main()

{

int val[15],wt[15],i,W,n;

printf("Enter the number of elements: ");

scanf("%d",&n);

printf("Enter the value:\n");

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

scanf("%d",&val[i]);

}

printf("Enter the weight:\n");

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

scanf("%d",&wt[i]);

}

printf("Enter the weight of bag: ");

scanf("%d",&W);

printf("\n Total Profit: %d", knapSack(W, wt, val, n));

getch();

}

Program 11(b)

#include <iostream>

#include<conio.h>

#include<stdio.h>

using namespace std;

typedef struct{

int v;

int w;

float d;

}Item;

void input(Item items[],int sizeOfItems)

{

cout << "Enter total "<< sizeOfItems <<" item's values and weight" << endl;

for(int i = 0; i < sizeOfItems; i++)

{

cout << "Enter "<< i+1 << " Value: ";

cin >> items[i].v;

cout << "Enter "<< i+1 << " value Weight: ";

cin >> items[i].w;

}

}

void display(Item items[], int sizeOfItems)

{

int i;

cout << "values: ";

for(i = 0; i < sizeOfItems; i++)

{

cout << items[i].v << "\t";

}

cout << endl << "weight: ";

for (i = 0; i < sizeOfItems; i++)

{

cout << items[i].w << "\t";

}

cout << endl;

}

bool compare(Item i1, Item i2)

{

return (i1.d > i2.d);

}

float knapsack(Item items[], int sizeOfItems, int W)

{

int i, j, pos;

Item mx, temp;

float totalValue = 0, totalWeight = 0;

for (i = 0; i < sizeOfItems; i++) {

items[i].d = items[i].v / items[i].w;

}

for(i=0; i<sizeOfItems; i++)

{

if(totalWeight + items[i].w<= W)

{

totalValue += items[i].v ;

totalWeight += items[i].w;

}

else

{

int wt = W-totalWeight;

totalValue += (wt \* items[i].d);

totalWeight += wt;

break;

}

}

cout << "total weight in bag:" << totalWeight<<endl;

return totalValue;

}

int main()

{

int W,n;

Item items[25];

cout << "Enter the number of elements: ";

cin >> n;

input(items, n);

cout << "Entered data: \n";

display(items,n);

cout<< "Enter Knapsack weight: \n";

cin >> W;

float mxVal = knapsack(items, n, W);

cout << "Max value for "<< W <<" weight is: "<< mxVal;

getch();

}

Program 12

Algorithm and  Program for job scheduling with deadline

Algorithm

1) Sort all jobs in decreasing order of profit.

2) Iterate on jobs in decreasing order of profit.For each job , do the following :

a)Find a time slot i, such that slot is empty and i < deadline and i is greatest.Put the job in

this slot and mark this slot filled.

b)If no such i exists, then ignore the job.

Program

#include<iostream>

#include<algorithm>

using namespace std;

struct Job

{

   char id;

   int dead;

   int profit;

};

bool comparison(Job a, Job b)

{

     return (a.profit > b.profit);

}

void printJobScheduling(Job arr[], int n)

{

    sort(arr, arr+n, comparison);

    int result[n];

    bool slot[n];

    for (int i=0; i<n; i++)

        slot[i] = false;

    for (int i=0; i<n; i++)

    {

       for (int j=min(n, arr[i].dead)-1; j>=0; j--)

       {

          if (slot[j]==false)

          {

             result[j] = i;

             slot[j] = true;

             break;

          }

       }

    }

    for (int i=0; i<n; i++)

       if (slot[i])

         cout << arr[result[i]].id << " ";

}

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]);

    cout << "Following is maximum profit sequence of jobs \n";

    printJobScheduling(arr, n);

    return 0;

}

Program 13

#include<iostream>

#include<vector>

using namespace std;

int assemblyLineScheduling(int n, vector<int> entry, vector<int> exit, vector<vector<int> > processing, vector<vector<int> > transfer)

{

vector<vector<int> > dp(2, vector<int>(n+1));

int i;

//initialization

//entry to first station

dp[0][0]=entry[0]+processing[0][0];

dp[1][0]=entry[1]+processing[1][0];

for(i=1;i<n;i++)

{

//for being on station i of assembly line 1

dp[0][i]=min(dp[0][i-1],dp[1][i-1]+transfer[1][i-1])+processing[0][i];

//for being on station i of assembly line 2

dp[1][i]=min(dp[1][i-1],dp[0][i-1]+transfer[0][i-1])+processing[1][i];

}

//exiting from the last station

dp[0][n]=dp[0][n-1]+exit[0];

dp[1][n]=dp[1][n-1]+exit[1];

return min(dp[0][n],dp[1][n]);

}

int main()

{

int i,n;

vector<int> entry(2), exit(2);

cout<<"Enter the number of stations ";

cin>>n;

vector<vector<int> > processing(2, vector<int> (n));

vector<vector<int> > transfer(2, vector<int> (n-1));

cout<<"Enter the entry time for assebly line 1 and 2 respectively"<<endl;

cin>>entry[0]>>entry[1];

cout<<"Enter the exit time for assebly line 1 and 2 respectively"<<endl;

cin>>exit[0]>>exit[1];

cout<<"Entry the processing time at all staions on assembly line 1"<<endl;

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

cin>>processing[0][i];

cout<<"Entry the processing time at all staions on assembly line 2"<<endl;

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

cin>>processing[1][i];

cout<<"Enter the transfer time from each station of assembly line 1 to next station of assembly line 2"<<endl;

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

cin>>transfer[0][i];

cout<<"Enter the transfer time from each station of assembly line 2 to next station of assembly line 1"<<endl;

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

cin>>transfer[1][i];

cout<<"The minimum time required to get all the jobs done is "<<endl;

cout<<assemblyLineScheduling(n, entry, exit, processing, transfer);

cout<<endl;

return 0;

}

Program 14

#include<iostream>

#include<limits.h>

using namespace std;

int MatrixChainMultiplication(int p[], int n)

{

int m[n][n];

int i, j, k, L, q;

for (i=1; i<n; i++)

m[i][i] = 0;

for (L=2; L<n; L++)

{

for (i=1; i<n-L+1; i++)

{

j = i+L-1;

m[i][j] = INT\_MAX;

for (k=i; k<=j-1; k++)

{

q = m[i][k] + m[k+1][j] + p[i-1]\*p[k]\*p[j];

if (q < m[i][j])

{

m[i][j] = q;

}

}

}

}

return m[1][n-1];

}

int main()

{

int n,i;

cout<<"Enter number of matrices\n";

cin>>n;

n++;

int arr[n];

cout<<"Enter dimensions \n";

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

{

cout<<"Enter d"<<i<<" :: ";

cin>>arr[i];

}

int size = sizeof(arr)/sizeof(arr[0]);

cout<<"Minimum number of multiplications is "<<MatrixChainMultiplication(arr, size);

return 0;

}

Program 15

#include<stdio.h>

#include<conio.h>

int max(int a, int b)

{

return (a > b) ? a : b;

}

int knapSack(int W, int wt[], int val[], int n)

{

if (n == 0 || W == 0)

return 0;

if (wt[n - 1] > W)

return knapSack(W, wt, val, n - 1);

else

return max(

val[n - 1] + knapSack(W - wt[n - 1], wt, val, n - 1),

knapSack(W, wt, val, n - 1));

}

void main()

{

int val[15],wt[15],i,W,n;

printf("Enter the number of elements: ");

scanf("%d",&n);

printf("Enter the value:\n");

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

scanf("%d",&val[i]);

}

printf("Enter the weight:\n");

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

scanf("%d",&wt[i]);

}

printf("Enter the weight of bag: ");

scanf("%d",&W);

printf("\n Total Profit: %d", knapSack(W, wt, val, n));

getch();

}