1) #include <stdio.h>

int main() {

int pages[] = {12, 34, 67, 90};

int N = 4;

int M = 2;

int total\_pages = 0;

int MAX\_VALUE = 1000000;

if (N < M) {

printf("-1\n");

return 0;

}

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

total\_pages += pages[i];

}

int low = 0, high = total\_pages;

int result = MAX\_VALUE;

while (low <= high) {

int mid = (low + high) / 2;

int requiredStudents = 1, currentSum = 0;

int feasible = 1;

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

if (pages[i] > mid) {

feasible = 0;

break;

}

if (currentSum + pages[i] > mid) {

requiredStudents++;

currentSum = pages[i];

if (requiredStudents > M) {

feasible = 0;

break;

}

} else {

currentSum += pages[i];

}

}

if (feasible) {

result = mid;

high = mid - 1;

} else {

low = mid + 1;

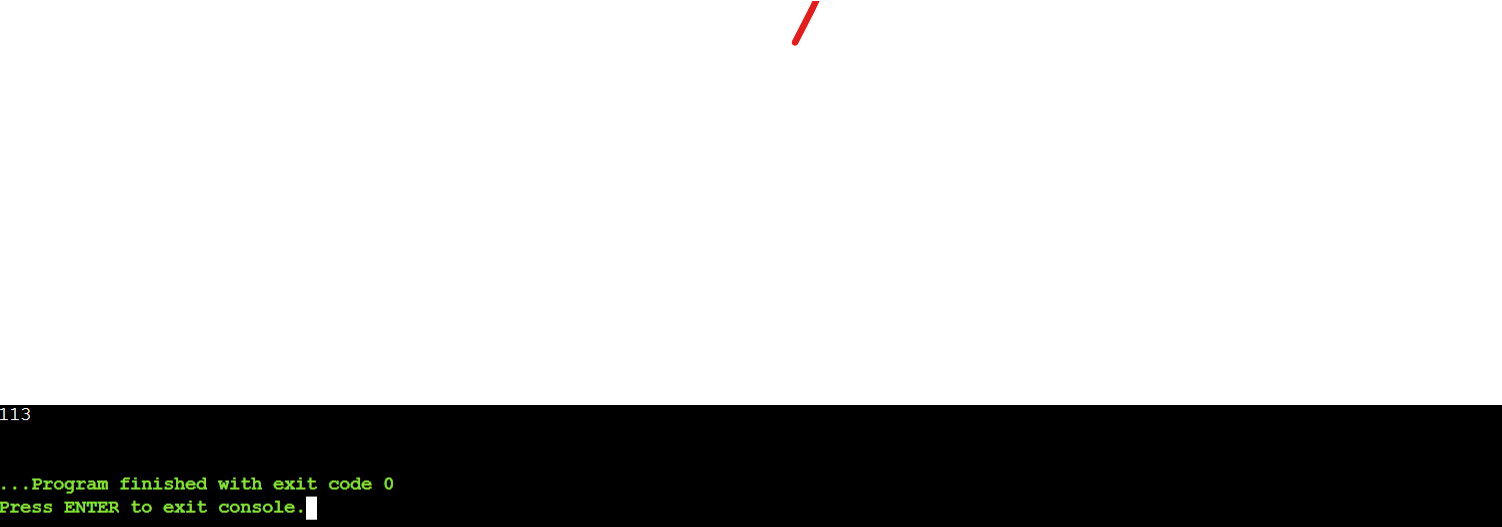
}

}

printf("%d\n", result);

return 0;

}



2)

#include <stdio.h>

int main() {

int numDenominations, targetAmount;

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

scanf("%d", &numDenominations);

int denominations[numDenominations];

printf("Enter the denominations: ");

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

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

}

printf("Enter the target amount: ");

scanf("%d", &targetAmount);

int INFINITY\_VALUE = 1000000;

int dp[targetAmount + 1];

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

dp[i] = INFINITY\_VALUE;

}

dp[0] = 0;

for (int currentAmount = 1; currentAmount <= targetAmount; currentAmount++) {

for (int j = 0; j < numDenominations; j++) {

if (denominations[j] <= currentAmount) {

int remainingAmount = dp[currentAmount - denominations[j]];

if (remainingAmount + 1 < dp[currentAmount]) {

dp[currentAmount] = remainingAmount + 1;

}

}

}

}

if (dp[targetAmount] == INFINITY\_VALUE) {

printf("-1\n");

} else {

printf("Minimum number of coins : %d\n", dp[targetAmount]);

}

return 0;

}

A black background with numbers

Description automatically generated

3)

#include <stdio.h>

#include <stdlib.h>

struct Node {

int value;

struct Node\* next;

};

int main() {

struct Node\* head = (struct Node\*)malloc(sizeof(struct Node));

struct Node\* second = (struct Node\*)malloc(sizeof(struct Node));

struct Node\* third = (struct Node\*)malloc(sizeof(struct Node));

struct Node\* fourth = (struct Node\*)malloc(sizeof(struct Node));

head->value = 1;

head->next = second;

second->value = 2;

second->next = third;

third->value = 3;

third->next = fourth;

fourth->value = 4;

fourth->next = second;

struct Node \*slowPointer = head, \*fastPointer = head;

struct Node \*previousNode = NULL;

int loopDetected = 0;

while (slowPointer && fastPointer && fastPointer->next) {

slowPointer = slowPointer->next;

fastPointer = fastPointer->next->next;

if (slowPointer == fastPointer) {

loopDetected = 1;

break;

}

}

if (loopDetected) {

slowPointer = head;

while (slowPointer != fastPointer) {

previousNode = fastPointer;

slowPointer = slowPointer->next;

fastPointer = fastPointer->next;

}

previousNode->next = NULL;

}

struct Node\* temp = head;

while (temp != NULL) {

printf("%d -> ", temp->value);

temp = temp->next;

}

printf("NULL\n");

free(head);

free(second);

free(third);

free(fourth);

return 0;

}

