**Task Assignment**

1. In this game, we have two players. In our program, player 1 will be the user, and player 2 will be the computer. Player 1 selects either rock, paper, or scissor. The computer does not know about what player 1 has selected, so the computer randomly selects any item (rock, paper, or scissor).

rock vs. scissors -> rock wins

paper vs. scissors -> scissors wins

paper vs. rock -> paper wins

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

int generateRandomNumber(int n)

{

srand(time(NULL)); //srand takes seed as an input and is defined inside stdlib.h

return rand() % n;

}

LOGIC ://Create Rock, Paper & Scissors Game

// Player 1: rock

// Player 2 (computer): scissors -->player 1 gets 1 point

// rock vs scissors - rock wins

// paper vs scissors - scissors wins

// paper vs rock - paper wins

int main()

{

printf("The random number between 0 to 5 is %d\n", generateRandomNumber(5));

return 0;

}

2. The first column has all values 0 because because the minimum number of coins to get change 0 is 0.

**Consider the first row:**

Cell(0,1) : The minimum number of coins to get change 1 when you can consider only coins of denomination 1 is 1 ({1}) .

Cell (0,2) : The minimum number of coins to get change 2 when you can consider only coins of denomination 1 is 2 ({1,1}).

So on and so forth till cell(0,11) in first row

**Consider the second row:**

Cells (1,0) …. (1,4) -> one can copy values from the cell directly above since you cannot get change using coins having denomination greater that the value of the change itself. (example — Cell(1,2) : get a change of 2 using coin of denomination 5 , hence we copy the solution from the cell directly above it which is Cell(0,2) . This means that the cell (1,2) holds the best possible solution to get a change 2 using coins of denominations 1 and 5)

Cell(1,5) : We can get the change 5 using a single coin of denomination 5 . Hence answer is 1.

Cell(1,6) : When we try to find a solution , we first take one coin of denomination 5. So, now we need to find the minimum number of coins required for 6–5=1. The solution for change 1 is given in Cell(1,1). Hence the answer is {1,5} and the minimum number of coins is 2.

In this way, we continue to fill the table row by row, from left to right.

In Cell(2,10) : To get change 10 using denomination 6 , one solution is {1,1,1,1,6} . This requires five coins. But this solution is not the best one. A better solution is {5,5} which has already been calculated in the cell above Cell(1,10).

// A Naive recursive C++ program to find minimum of coins

// to make a given change V

ANSWER:

#include<bits/stdc++.h>

using namespace std;

int minCoins(int coins[], int m, int V)

{

if (V == 0) return 0;

int res = INT\_MAX;

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

{

if (coins[i] <= V)

{

int sub\_res = minCoins(coins, m, V-coins[i]);

if (sub\_res != INT\_MAX && sub\_res + 1 < res)

res = sub\_res + 1;

}

}

return res;

}

int main()

{

int coins[] = {1, 2, 5, 10};

int m = sizeof(coins)/sizeof(coins[0]);

int V = 11;

cout << "Minimum coins required is "

<< minCoins(coins, m, V);

return 0;

}

3.

A simple rule for using numbers in writing is that small numbers ranging from one to ten (or one to nine, depending on the style guide) should generally be spelled out. Larger numbers (i.e., above ten) are written as numerals.

Answer:

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

void convert\_to\_words(char\* num)

{

int len = strlen(

num);

if (len == 0) {

fprintf(stderr, "empty string\n");

return;

}

if (len > 4) {

fprintf(stderr,

"Length more than 4 is not supported\n");

return;

}

char\* single\_digits[]

= { "zero", "one", "two", "three", "four",

"five", "six", "seven", "eight", "nine" };

char\* two\_digits[]

= { "", "ten", "eleven", "twelve",

"thirteen", "fourteen", "fifteen", "sixteen",

"seventeen", "eighteen", "nineteen" };

char\* tens\_multiple[] = { "", "", "twenty",

"thirty", "forty", "fifty",

"sixty", "seventy", "eighty",

"ninety" };

char\* tens\_power[] = { "hundred", "thousand" };

printf("\n%s: ", num);

if (len == 1) {

printf("%s\n", single\_digits[\*num - '0']);

return;

}

while (\*num != '\0') {

if (len >= 3) {

if (\*num - '0' != 0) {

printf("%s ", single\_digits[\*num - '0']);

printf("%s ",

tens\_power[len - 3]);

}

--len;

}

else {

if (\*num == '1') {

int sum = \*num - '0' + \*(num + 1) - '0';

printf("%s\n", two\_digits[sum]);

return;

}

else if (\*num == '2' && \*(num + 1) == '0') {

printf("twenty\n");

return;

}

else {

int i = \*num - '0';

printf("%s ", i ? tens\_multiple[i] : "");

++num;

if (\*num != '0')

printf("%s ",

single\_digits[\*num - '0']);

}

}

++num;

}

}

int main(void)

{

convert\_to\_words("9923");

convert\_to\_words("523");

convert\_to\_words("89");

convert\_to\_words("8");

return 0;

}