Practical – 10

Aim :- Implement Program for "Making Change" using Dynamic Programming .

Code:-

```
#include <stdio.h>
#include <stdlib.h>


struct change_entry { unsigned int count; int coin;
struct change_entry *prev;

};

typedef struct change_entry change_entry;


unsigned int make_change(const unsigned int *coins, size_t len, unsigned int value,
unsigned int **solution)

{

unsigned int i, j; change_entry **table; unsigned int result;


/* Allocate the table */

table = malloc((len + 1) * sizeof(change_entry *)); for (i = 0; i <= len;
i++) { table[i] = malloc((value + 1) * sizeof(change_entry));

}


/* Calculate the values and build chains */ for (i = 0; i <=
len; i++) { for (j = 0; j <= value; j++) { if (i == 0) { /*
Initialising the first row */ table[i][j].count = j;
table[i][j].coin = -1;  s
table[i][j].prev = NULL;
```

```
}

else if (j == 0) {

/* Initialising the first column */ table[i][j].count = 0;
table[i][j].coin = -1;
table[i][j].prev = NULL;
}

else if (coins[i - 1] == j) {

/* Can use coin */

table[i][j].count = 1;
table[i][j].coin = i - 1; table[i][j].prev = NULL;
}

else if (coins[i - 1] > j) {

/* Can't use coin */

table[i][j].count = table[i - 1][j].count;
table[i][j].coin = -1;
table[i][j].prev = &table[i - 1][j];

}

else {

/* Can use coin - choose best solution */

if (table[i - 1][j].count < table[i][j - coins[i - 1]].count + 1) {

/* Don't use coin */
```

```
table[i][j].count = table[i - 1][j].count; table[i][j].coin = -1;
table[i][j].prev = &table[i - 1][j];

}

else {

/* Use coin */

table[i][j].count = table[i][j - coins[i - 1]].count + 1;
table[i][j].coin = i - 1;

table[i][j].prev = &table[i][j - coins[i - 1]];

}

}

}

}

result = table[len][value].count;

/* Read back the solution */

*solution = calloc(len, sizeof(unsigned int));
if (*solution) {
change_entry *head;

for (head = &table[len][value]; head != NULL;
head = head->prev) { if (head->coin != -1) {
(*solution)[head->coin]++;

}
```

```
}

}

else {

result = 0;

}

for (i = 0; i <= len; i++) { free(table[i]);
}

free(table);


return result;

}

int main(void)

{

unsigned int coins[] = {1, 2, 5, 10, 20, 50, 100}; const size_t len = sizeof(coins) /
sizeof(coins[0]); const unsigned int value = 252;
unsigned int *solution;

unsigned int result = make_change(coins, len, value, &solution);
unsigned int i; printf("Number of coins: %u\n", result);  printf("Coins
used:\n");
for (i = 0; i < len; i++) { if (solution[i] > 0) {
printf("%u x %u\n", solution[i], coins[i]);

}

}
```

```
free(solution); return 0;
}
```

Output:-