# Software Engineering (CS401)

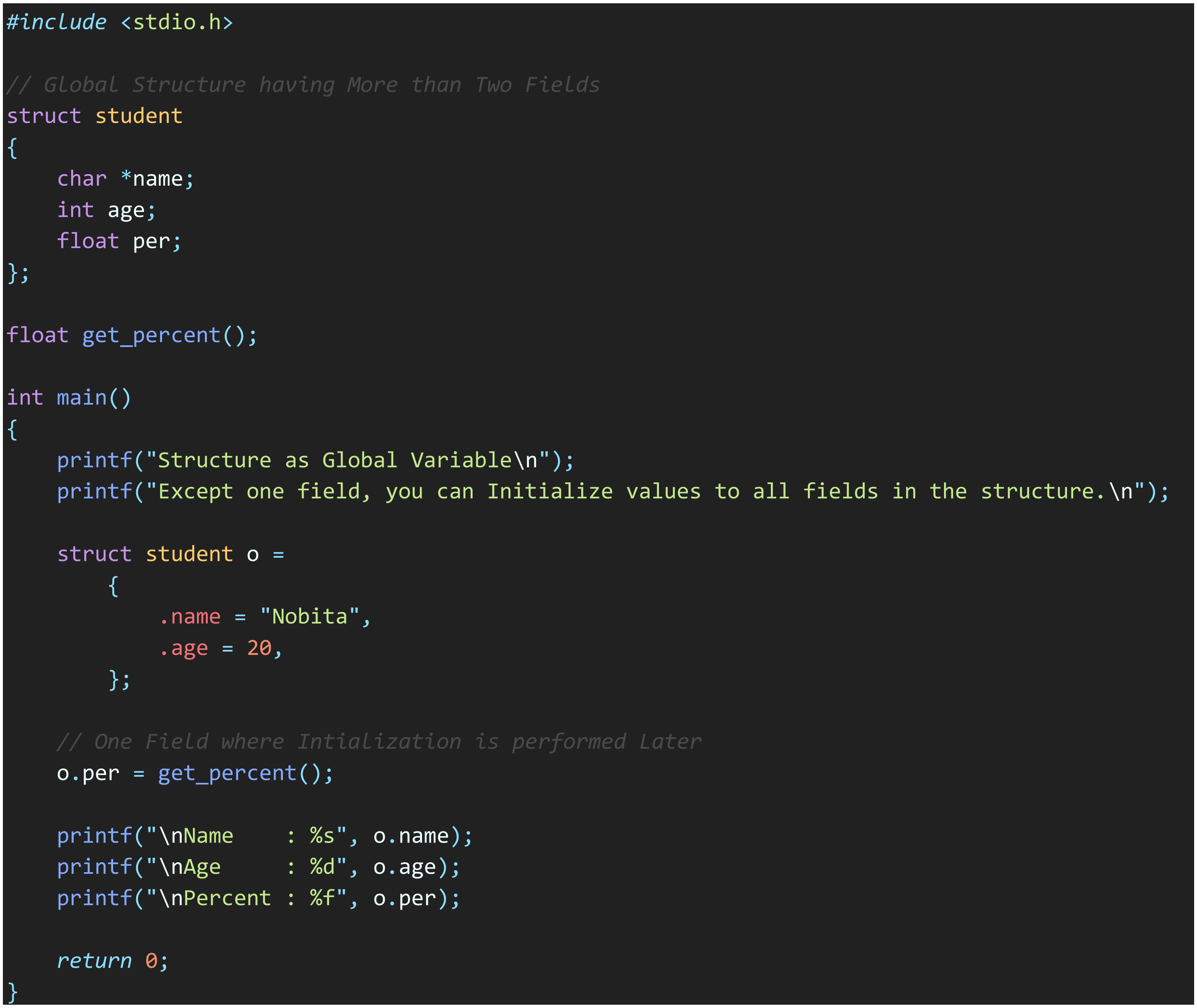
## Practical Lab Examination

## U19CS012

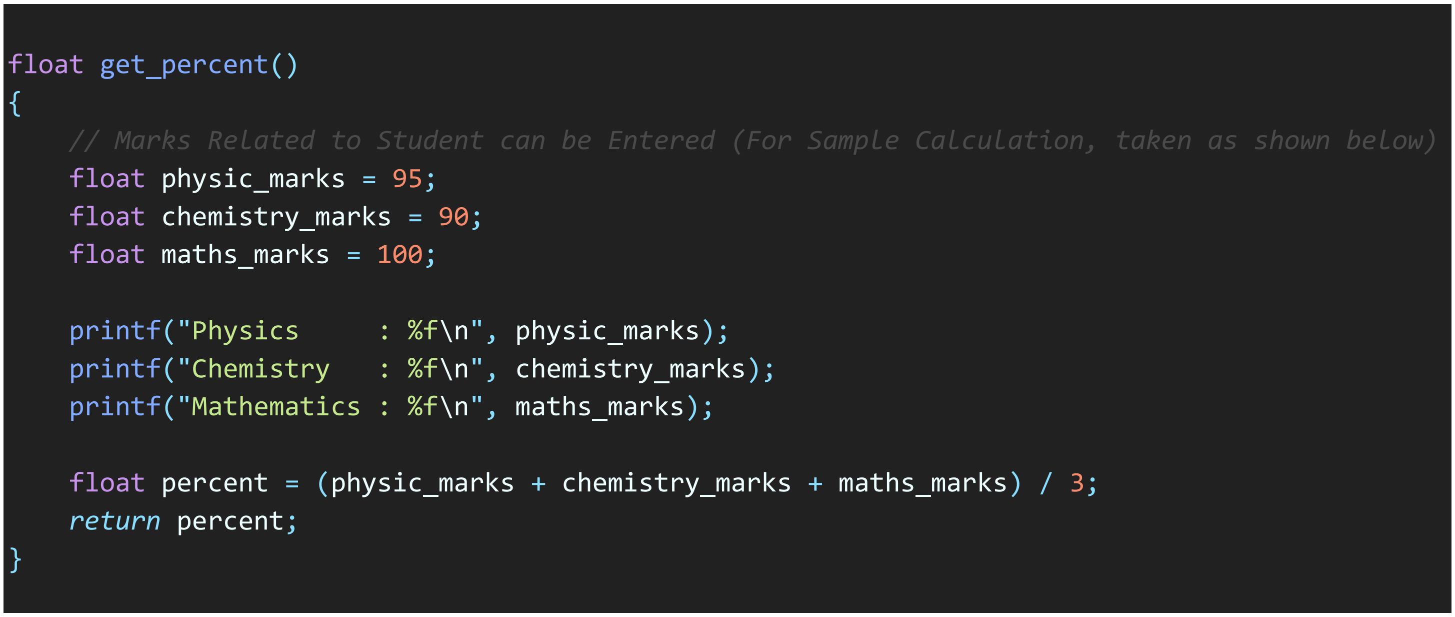
Q1.)

a.) Write a C program having structure as global variable. This structure can have more than two fields. Except for one field, you can initialize values to all fields in the structure. Run Splint for this C code and report the error generated. (This code should generate error as you have one uninitialized field in structure)

### Code

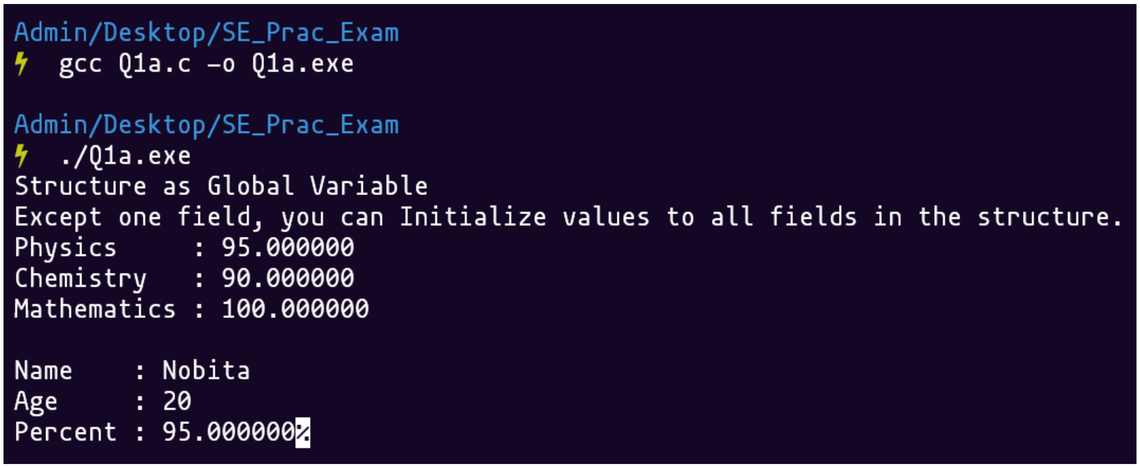


```c
#include <stdio.h>

// Global Structure having More than Two Fields
struct student
{
    char *name;
    int age;
    float per;
};

float get_percent();

int main()
{
    printf("Structure as Global Variable\n");
    printf("Except one field, you can Initialize values to all fields in the structure.\n");

    struct student o =
        {
            .name = "Nobita",
            .age = 20,
        };

    // One Field where Intialization is performed Later
    o.per = get_percent();

    printf("\nName    : %s", o.name);
    printf("\nAge     : %d", o.age);
    printf("\nPercent : %f", o.per);

    return 0;
}
```

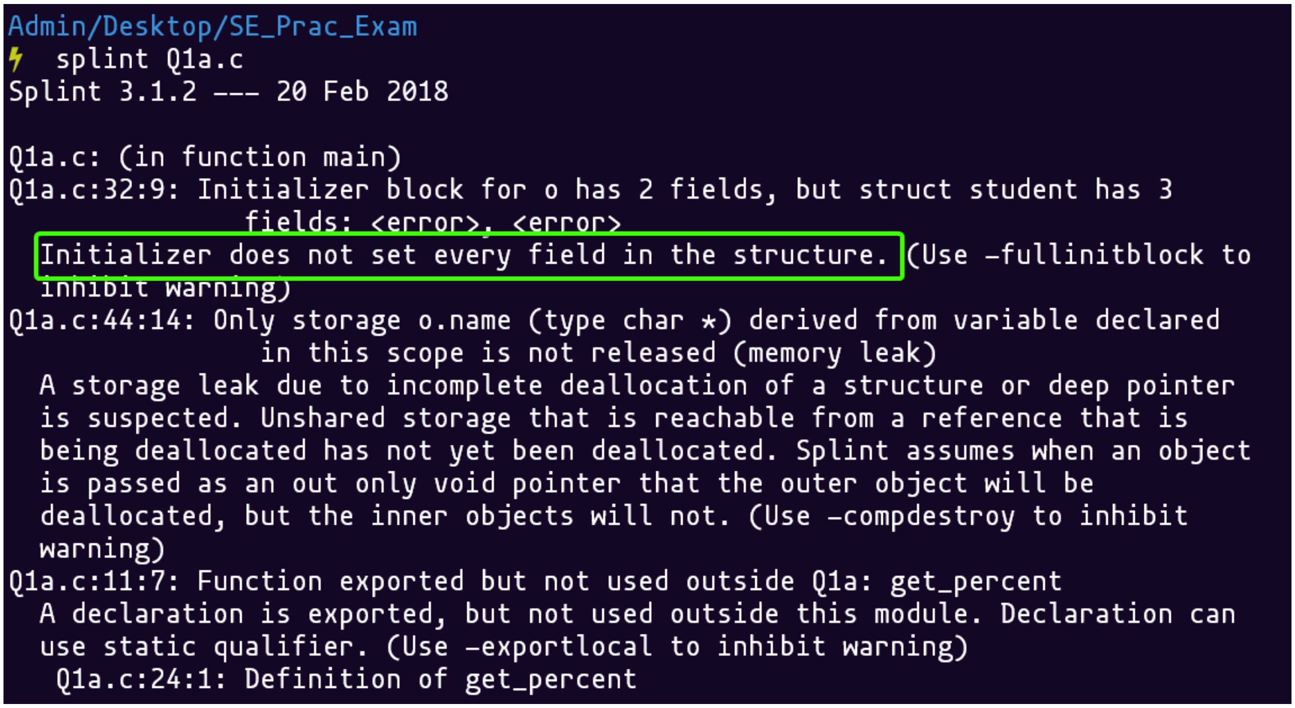


```c
float get_percent()
{
    // Marks Related to Student can be Entered (For Sample Calculation, taken as shown below)
    float physic_marks = 95;
    float chemistry_marks = 90;
    float maths_marks = 100;

    printf("Physics     : %f\n", physic_marks);
    printf("Chemistry   : %f\n", chemistry_marks);
    printf("Mathematics : %f\n", maths_marks);

    float percent = (physic_marks + chemistry_marks + maths_marks) / 3;
    return percent;
}
```

## Output



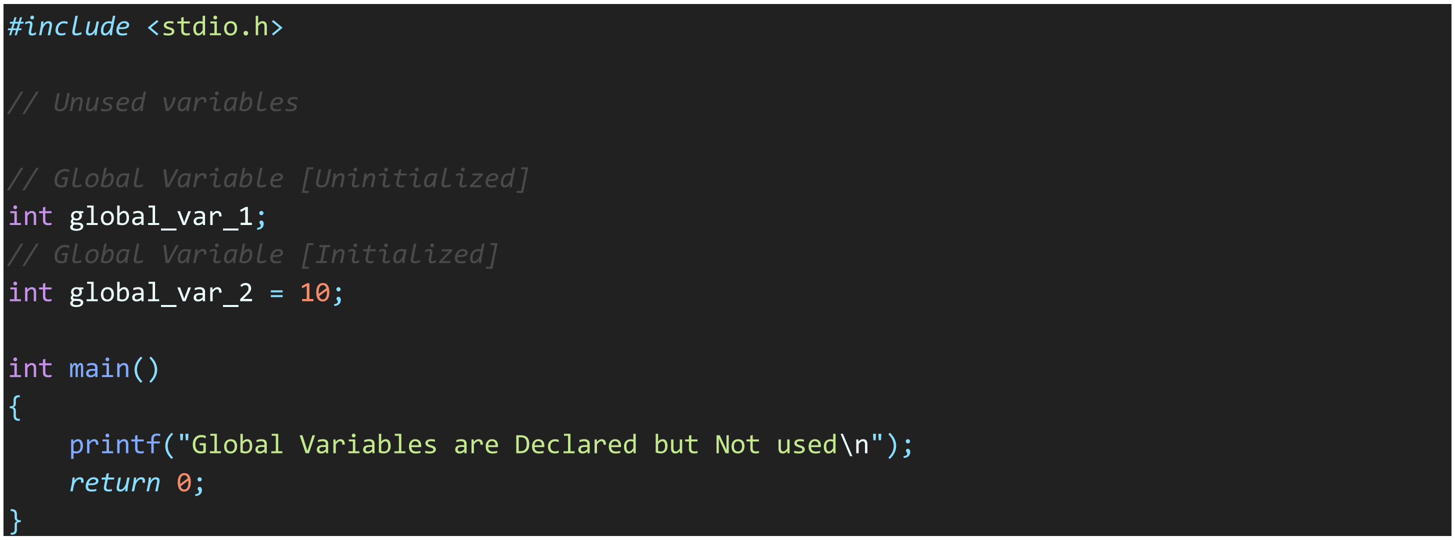
```
Admin/Desktop/SE_Prac_Exam
ϟ  gcc Q1a.c -o Q1a.exe

Admin/Desktop/SE_Prac_Exam
ϟ  ./Q1a.exe
Structure as Global Variable
Except one field, you can Initialize values to all fields in the structure.
Physics     : 95.000000
Chemistry   : 90.000000
Mathematics : 100.000000

Name    : Nobita
Age     : 20
Percent : 95.000000%
```



```
Admin/Desktop/SE_Prac_Exam
ϟ  splint Q1a.c
Splint 3.1.2 --- 20 Feb 2018

Q1a.c: (in function main)
Q1a.c:32:9: Initializer block for o has 2 fields, but struct student has 3
               fields: <error>, <error>
  Initializer does not set every field in the structure. (Use -fullinitblock to
  inhibit warning)
Q1a.c:44:14: Only storage o.name (type char *) derived from variable declared
                in this scope is not released (memory leak)
  A storage leak due to incomplete deallocation of a structure or deep pointer
  is suspected. Unshared storage that is reachable from a reference that is
  being deallocated has not yet been deallocated. Splint assumes when an object
  is passed as an out only void pointer that the outer object will be
  deallocated, but the inner objects will not. (Use -compdestroy to inhibit
  warning)
Q1a.c:11:7: Function exported but not used outside Q1a: get_percent
  A declaration is exported, but not used outside this module. Declaration can
  use static qualifier. (Use -exportlocal to inhibit warning)
   Q1a.c:24:1: Definition of get_percent
```

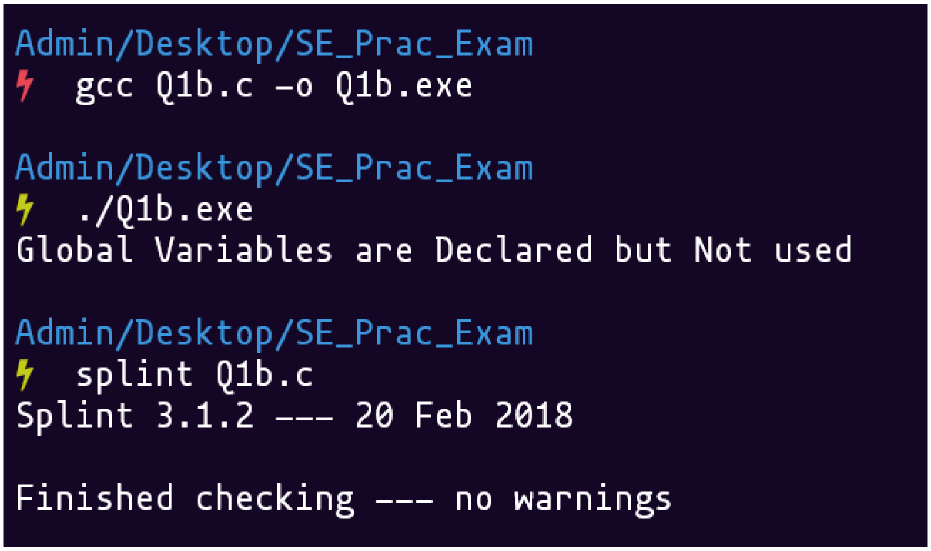
b.) Write a C program having some global variables that are declared but not used anywhere in the code. Run Splint for this C code and report the error generated.

## Code



```c
#include <stdio.h>

// Unused variables

// Global Variable [Uninitialized]
int global_var_1;
// Global Variable [Initialized]
int global_var_2 = 10;

int main()
{
    printf("Global Variables are Declared but Not used\n");
    return 0;
}
```

## Output



```
Admin/Desktop/SE_Prac_Exam
⚡  gcc Q1b.c -o Q1b.exe

Admin/Desktop/SE_Prac_Exam
⚡  ./Q1b.exe
Global Variables are Declared but Not used

Admin/Desktop/SE_Prac_Exam
⚡  splint Q1b.c
Splint 3.1.2 --- 20 Feb 2018

Finished checking --- no warnings
```

Q2.) Consider a chocolate vending machine that contains **Four** types of chocolates

| **Chocolate Type** | **Price** |
|---|---|
| Coffeestick | Rs 10 |
| Milkbar | Rs 20 |
| Silkchocolate | Rs 50 |
| DarkChocolate | Rs 50 |

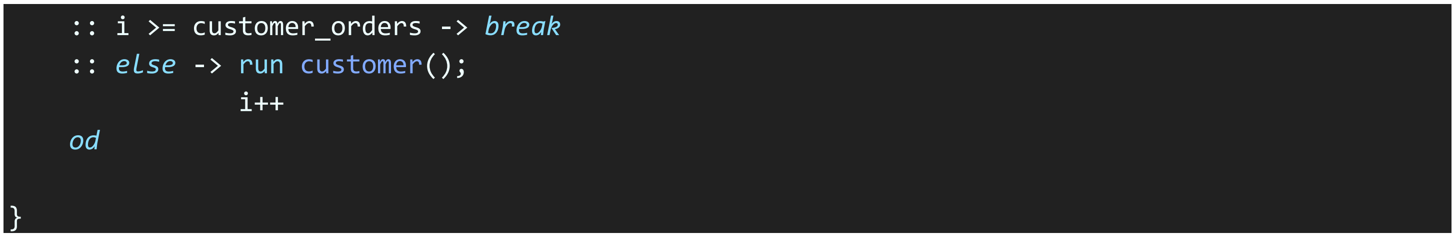
To select a coffeestick, one inserts 10 rupee note. This process is similar with other types of chocolate. For example, to collect a milkbar, one has to insert 20 rupee note.

Create a SPIN model of these **two processes**, i.e., vender for vending machine having limitless supply and **customer** having limitless appetite for chocolates.

You model will use message channels for communication between customer and vender. In the second step, you will set a limit for vending machine, i.e., at most 20 coffeesticks, 20 milkbars, 10 silkchocolates and 15 darkchocolates can be supplied in a single run (after this, vending machine needs to refill). You program should print the amount collected by vending machine at the time where it needs refill or no more chocolate (of any type) can be provided thereafter.

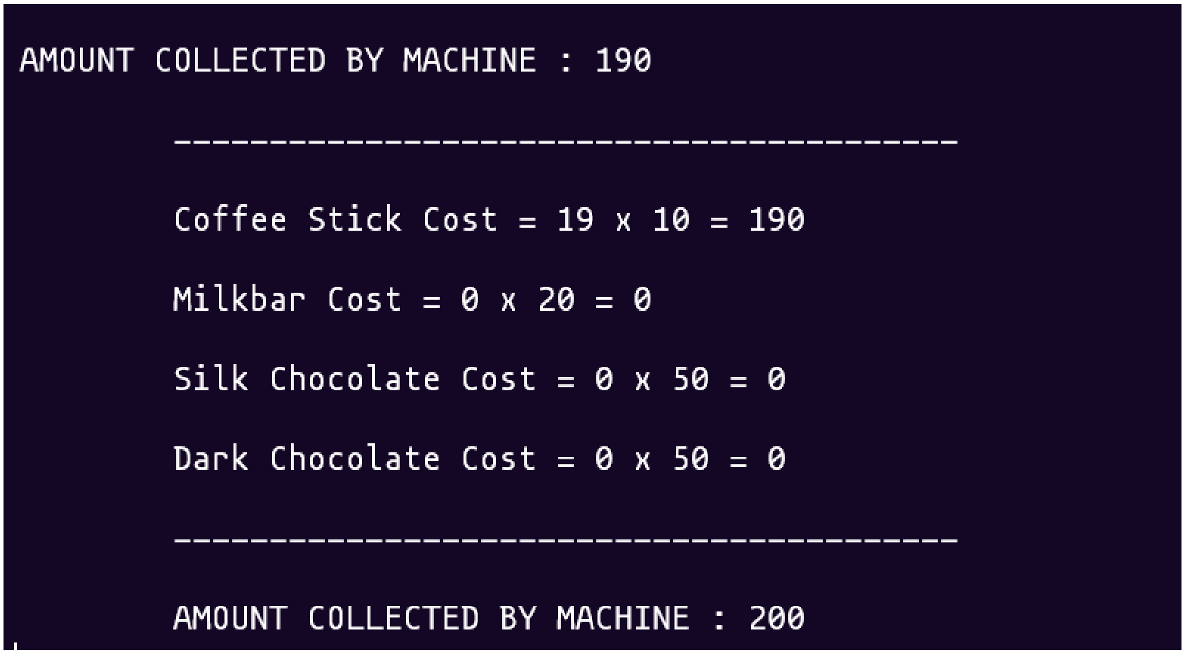
## Code

```
#define MAX_SUPPLY 10000

// chan STDIN = [4] of int;

int coffeestick_available = MAX_SUPPLY;
int milkbars_available = MAX_SUPPLY;
int silkchocolates_available = MAX_SUPPLY;
int darkchocolates_available = MAX_SUPPLY;

int coffeestick_cost = 10;
int milkbar_cost = 20;
int silkchocolate_cost = 50;
int darkchocolate_cost = 50;

int coffeestick_count = 0;
int milkbar_count = 0;
int silkchocolate_count = 0;
int darkchocolate_count = 0;
int total_amount = 0;

proctype print_amount_collected()
{
    // do
    // ::
    printf("Chocolate Vending Machine Needs to be Refilled!\n\n");
    printf("AMOUNT COLLECTED BY MACHINE\n\n");
    printf("-----------------------------------------\n");
    printf("Coffee Stick Cost = %d x %d = %d\n", coffeestick_count, coffeestick_cost,
coffeestick_count*coffeestick_cost);
    printf("Milkbar Cost = %d x %d = %d\n", milkbar_count, milkbar_cost,
milkbar_count*milkbar_cost);
    printf("Silk Chocolate Cost = %d x %d = %d\n", silkchocolate_count, silkchocolate_cost,
silkchocolate_count*silkchocolate_cost);
    printf("Dark Chocolate Cost = %d x %d = %d\n", darkchocolate_count, darkchocolate_cost,
darkchocolate_count*darkchocolate_cost);
    printf("-----------------------------------------\n");
    printf("AMOUNT COLLECTED BY MACHINE : %d\n", total_amount);
    // break;
    // od
}

proctype vender(int id)
{
    // Having Limitless Supply

    // Having Limited Supply
    assert(id<=4);

    do
    :: printf("Vender start\n");
    od

    if
    :: (coffeestick_count == coffeestick_available) -> run  print_amount_collected();
    :: else -> if
               :: (milkbar_count == milkbars_available) -> run  print_amount_collected();
               :: else -> if
                          :: (silkchocolate_count == silkchocolates_available) ->
run  print_amount_collected();
                          :: else if
                                  :: (darkchocolate_count<darkchocolates_available) -> run
print_amount_collected();
                                  fi
                          fi
                fi
    fi

    assert(coffeestick_count<coffeestick_available);
    assert(milkbar_count<milkbars_available);
    assert(silkchocolate_count<silkchocolates_available);
    assert(darkchocolate_count<darkchocolates_available);

    if
    :: (id == 1) -> coffeestick_count = coffeestick_count + 1; total_amount = total_amount + 
coffeestick_cost;
    :: else -> if
                :: (id == 2) -> milkbar_count = milkbar_count + 1; total_amount = 
total_amount + milkbar_cost;
                else -> if
                        :: (id == 3) -> silkchocolate_count = silkchocolate_count + 1; 
total_amount = total_amount + silkchocolate_cost;
                        :: else -> darkchocolate_count = darkchocolate_count + 1; 
total_amount = total_amount + darkchocolate_cost;
                        fi
               fi
    fi

}

proctype customer()
{
    printf("Customer start\n");
    // Customer Needs to Enter the Order
    printf("Enter your Choice\n 1 - coffeestick 2 - milkbar 3 - silkchocolates 4 - 
darkchocolates\n");
    int customer_choice = 1;
    // cin >> customer_choice;
    run vender(customer_choice);
}

init
{
    printf("Welcome to Chocolate Vending Machine.\n\n");

    // This is Boolean flag for Part - 2
    bool has_limited_supply = false;
    has_limited_supply = true;
    if
    :: (has_limited_supply == true) -> coffeestick_available = 20; milkbars_available = 20; 
silkchocolates_available = 10; darkchocolates_available = 15;
    // :: else -> break;
    fi

    int customer_orders = 21;

    int i = 0;

    do
```

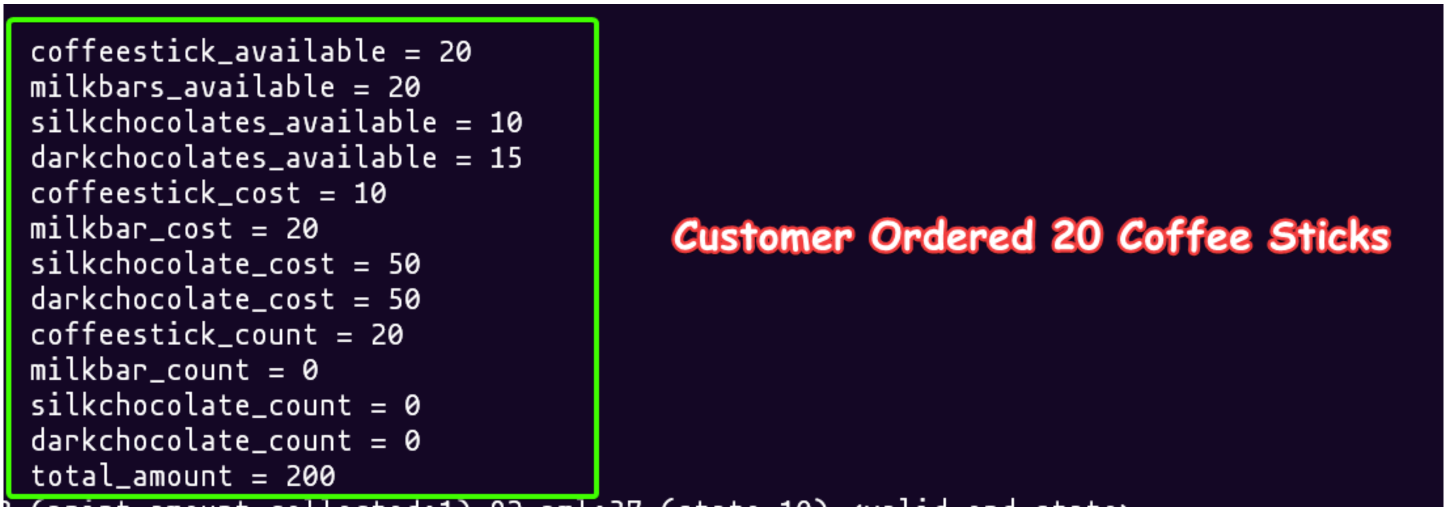


```
    :: i >= customer_orders -> break
    :: else -> run customer();
              i++
    od

}
```

## Output



```
AMOUNT COLLECTED BY MACHINE : 190

        ----------------------------------------

        Coffee Stick Cost = 19 x 10 = 190

        Milkbar Cost = 0 x 20 = 0

        Silk Chocolate Cost = 0 x 50 = 0

        Dark Chocolate Cost = 0 x 50 = 0

        ----------------------------------------

        AMOUNT COLLECTED BY MACHINE : 200
```



```
coffeestick_available = 20
milkbars_available = 20
silkchocolates_available = 10
darkchocolates_available = 15
coffeestick_cost = 10
milkbar_cost = 20
silkchocolate_cost = 50
darkchocolate_cost = 50
coffeestick_count = 20
milkbar_count = 0
silkchocolate_count = 0
darkchocolate_count = 0
total_amount = 200
```

SUBMITTED BY:

U19CS012

BHAGYA VINOD RANA