

Development of <FSM Based Vending Machine> for <Model of Computation>

Course Name: Theory of Computation

Course Code: CSC 397

Section: B

A Design of Vending Machine (FSM) Report Submitted by:

| SL | ID | Name |
|----|----------|----------------|
| 29 | 20103167 | Md. Min Khayer |

Submitted to:

Krishna Das,

Asst. Professor,



Department of Computer Science and Engineering

College of Engineering and Technology

IUBAT – International University of Business Agriculture and Technology

Spring 2023

ACKNOWLEDGMENTS

First and foremost, praise and thanks to the Almighty, for His showers of blessings throughout my work to complete the Theory of Computation course successfully.

I would like to express deep and sincere gratitude to my Course Instructor, Assistant Prof. Krishna Das for giving me the opportunity to do this assignment and for the continuous support, motivation, enthusiasm, and immense knowledge. His guidance helped me throughout the course and writing of this report. I could not have imagined having better course instructor and mentor for doing this course.

Last but not least, I would like to thank our parents, friends and respondents who have helped me with their valuable suggestions and guidance has been very helpful in various phases of the completion of this course.

ABSTRACT

Automata theory and design are core areas of most of the software developments which improves the efficiency of the vending machines used in all sectors including academic environments. In this problem involves designing a Finite State Machine (FSM) and a program to issue tickets for the Dhaka Metro Rail system in Bangladesh. The system offers single-use tickets and long-term MRT passes, with the MRT passes available for purchase at metro station counters. For the first phase of the project, passengers can purchase single-journey tickets from ticket vending machines between two stations, Uttara North to Agargaon. The program should accept input banknotes and issue appropriate messages along with the ticket issuance, based on three scenarios - insufficient amount, exact amount, and higher amount than the ticket fare. The FSM should be in a rejected or acceptance state, depending on the input amount.

Table of Contents

| | |
|---|----|
| ACKNOWLEDGMENTS | i |
| ABSTRACT..... | ii |
| Chapter I..... | 1 |
| 1.1 Finite Automata..... | 1 |
| 1.1.1 DFA (Deterministic Finite Automata) | 1 |
| Chapter II. | 3 |
| 2.1 Problem Analysis | 3 |
| 2.1.1 Input symbol and possible strings | 3 |
| 2.1.2 Accept and reject strings..... | 3 |
| Chapter III. | 4 |
| 3.1 Design the FSM (Vending Machine) | 4 |
| 3.2 FSM (Vending Machine) working process | 4 |
| 3.3 Transition Table of the FSM (Vending Machine)..... | 5 |
| Chapter IV..... | 6 |
| 4.1 Implementing the design through a programming language..... | 6 |
| Chapter V..... | 7 |
| 5.1 Source code of implementing FSM..... | 7 |
| Chapter VI..... | 9 |
| 6.1 The output of the program of vending machine..... | 9 |

Chapter I.

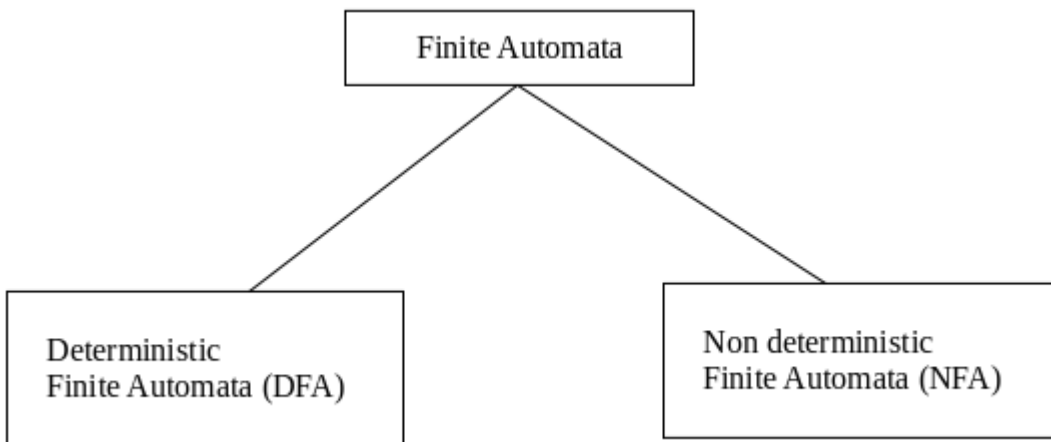
1.1 Finite Automata

Finite automata or finite state machine is the simplest machine that is used for recognizing patterns. It is an abstract machine that has five elements or tuples. It possesses a set of states and rules that are used for moving from one state to another, but it is dependent on the applied input symbol. The machine accepts the input if it is in the accepting state at the end of the string; otherwise, it rejects the input.

Types of Automata:

There are two types of finite automata:

1. DFA (deterministic finite automata)
2. NFA (non-deterministic finite automata)



1.1.1 DFA (Deterministic Finite Automata)

Deterministic refers to the uniqueness of the computation. In the DFA, the machine goes to one state only for a particular input character. DFA does not accept the null move.

A deterministic finite automaton is a set of five tuples. It is represented as:

$$M = (Q, \Sigma, \delta, q_0, F)$$

Where, Q : set of all states.

Σ : set of input symbols. (Symbols which machine takes as input)

q_0 : Initial state. (Starting state of a machine)

F : set of final state.

δ : Transition Function, defined as $\delta: Q \times \Sigma \rightarrow Q$.

For example, below DFA with $\Sigma = \{0, 1\}$ accepts all strings ending with 0.

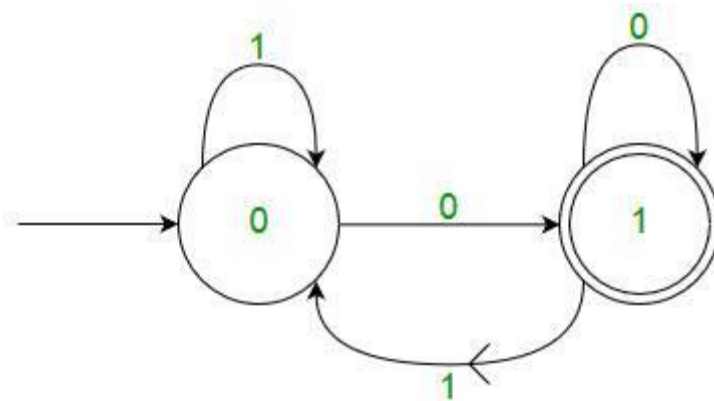


Figure 1.1: DFA with $\Sigma = \{0, 1\}$

List of DFA Applications:

- Vending Machines
- Video Games
- Text Parsing
- CPU Controllers
- Natural Language Processing
- Speech Recognition

1.1.2 NFA (Non-deterministic Finite Automata): It is used to transmit any number of states for a particular input. It can accept Null (or ϵ) move is allowed i.e., it can move forward without reading symbols.

For example, below NFA with $\Sigma = \{0, 1\}$ accepts all strings ending with 0.

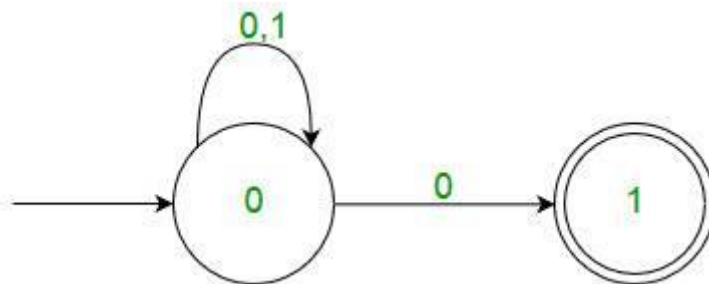


Figure 1.2: NFA with $\Sigma = \{0, 1\}$

Chapter II.

2.1 Problem Analysis

In the given problem vending machine allowed all Bangladeshi banknotes up to 100 taka note. Such as: 1, 2, 5, 10, 20, 50, 100.

The Fare has been decided in between Uttara North to Agargaon stations are 60 Bangladeshi Taka. The machine will not issue any ticket as long as it is not being received at least 60 taka. So, any combination of the string of banknote summation is less than 60 taka then it could be always rejected states and could not issue any ticket it is showing the insufficient amount message. Whenever the machine is being received 60 taka then it could be accepted states and issue a ticket. Once the machine will receive 60 taka or more then no other banknotes will be accepted that means from the accepted states input transition not allowed. Again If the machine is being received more than 60 taka then it could be accepted states, issue a ticket and return the remaining amount (Input Amount-60). Based on this problem logic have to design a vending machine FSM and write the program as well. For designing the FSM have to consider all the Bangladeshi Banknotes up to 100 taka note as a input symbols. Also consider all of these banknotes as a string then sum up the string. Check whether the sum of the string is 60 taka or not. After that apply those three logic and based on these design the vending machine (FSM).

2.1.1 Input symbol and possible strings

In this given problem the Finite State Machine input symbols are $\Sigma = \{1, 2, 5, 10, 20, 50, 100\}$

For making the design simple have to consider the input symbols are $\Sigma = \{10, 20, 50, 100\}$ and the possible strings are any combination of these symbols. As like: 10 20 50, 10 10 10 10 10 10, 10 20 10 10 10, 10 10 10 20 10.

2.1.2 Accept and reject strings

If the string summation less than 60 taka then the string is rejected. If the string summation is equal or more than 60 taka then the string is accepted by the FSM. For example: 10 20 10 10 10

Sum of the string= 10+20+10+10+10= 60 (Accepted String)

| Input String | States |
|-------------------|----------|
| 10 | Rejected |
| 10 10 | Rejected |
| 10 10 10 | Rejected |
| 10 10 10 10 10 10 | Accepted |
| 10 20 10 10 10 | Accepted |
| 20 50 | Accepted |
| 100 | Accepted |

Chapter III.

3.1 Design the FSM (Vending Machine)

Input symbol, $\Sigma = \{10, 20, 50, 100\}$

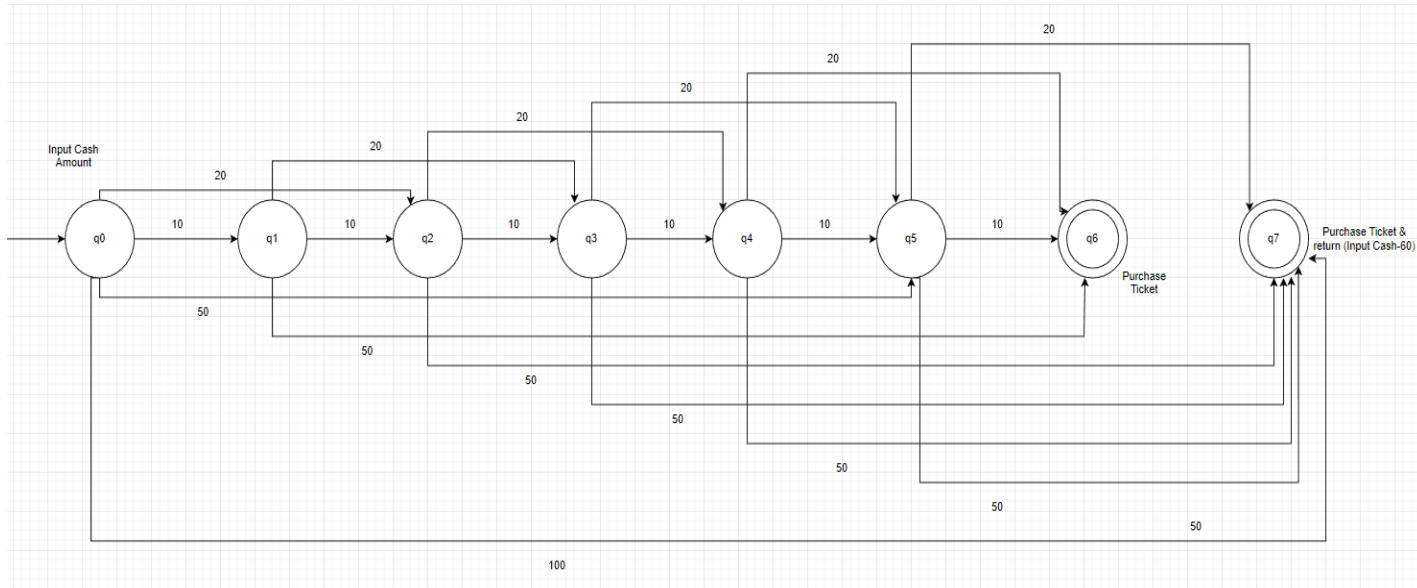
$$L = \{10\ 10\ 10\ 10\ 10\ 10, 10\ 20\ 10\ 10\ 10, 10, 10\ 10\ 20\ 10\ 10, 10\ 20\ 20\ 10, 20\ 50\ \dots\dots\dots\}$$


Figure 3.1: Design FSM of Vending Machine

3.2 FSM (Vending Machine) working process

The vending machine is first turned on, it initializes itself to a default state, which is typically the "idle" state. There are total 8 states of this FSM. The q0, q1, q2, q3, q4, q5 are rejected state or insufficient amount states and q6 is accepted states. Also q7 is accepted state but it returns remaining amount after purchase a ticket compare with the ticket fare amount. When a banknote is entered, the FSM checks if the total amount is less than the ticket fare. If yes, it moves to the Insufficient state, where further inputs can be accepted. If the total amount becomes equal to the ticket fare, the FSM moves to the Exact acceptance state q6, and no further inputs are allowed. If the total amount becomes more than the ticket fare, the FSM moves to the Excess acceptance state q7, where the remaining amount is returned along with the ticket.

The main logic is FSM can remain in the Rejected state, waiting for more inputs until the total amount becomes sufficient for the ticket fare, or the user cancels the transaction. Once the FSM reaches the Acceptance state, the ticket is issued, and the transaction is complete.

3.3 Transition Table of the FSM (Vending Machine)

| δ | 10 | 20 | 50 | 100 |
|----------|--------|--------|--------|--------|
| q0 | Q1 | Q2 | Q5 | Q7 |
| q1 | Q2 | Q3 | Q6 | ϕ |
| q2 | Q3 | Q4 | Q7 | ϕ |
| q3 | Q4 | Q5 | Q7 | ϕ |
| q4 | Q5 | Q6 | Q7 | ϕ |
| q5 | Q6 | Q7 | Q7 | ϕ |
| q6 | ϕ | ϕ | ϕ | ϕ |
| q7 | ϕ | ϕ | ϕ | ϕ |

Another design of FSM (Vending Machine) with Input symbol, $\Sigma = \{5, 10, 20, 50, 100\}$

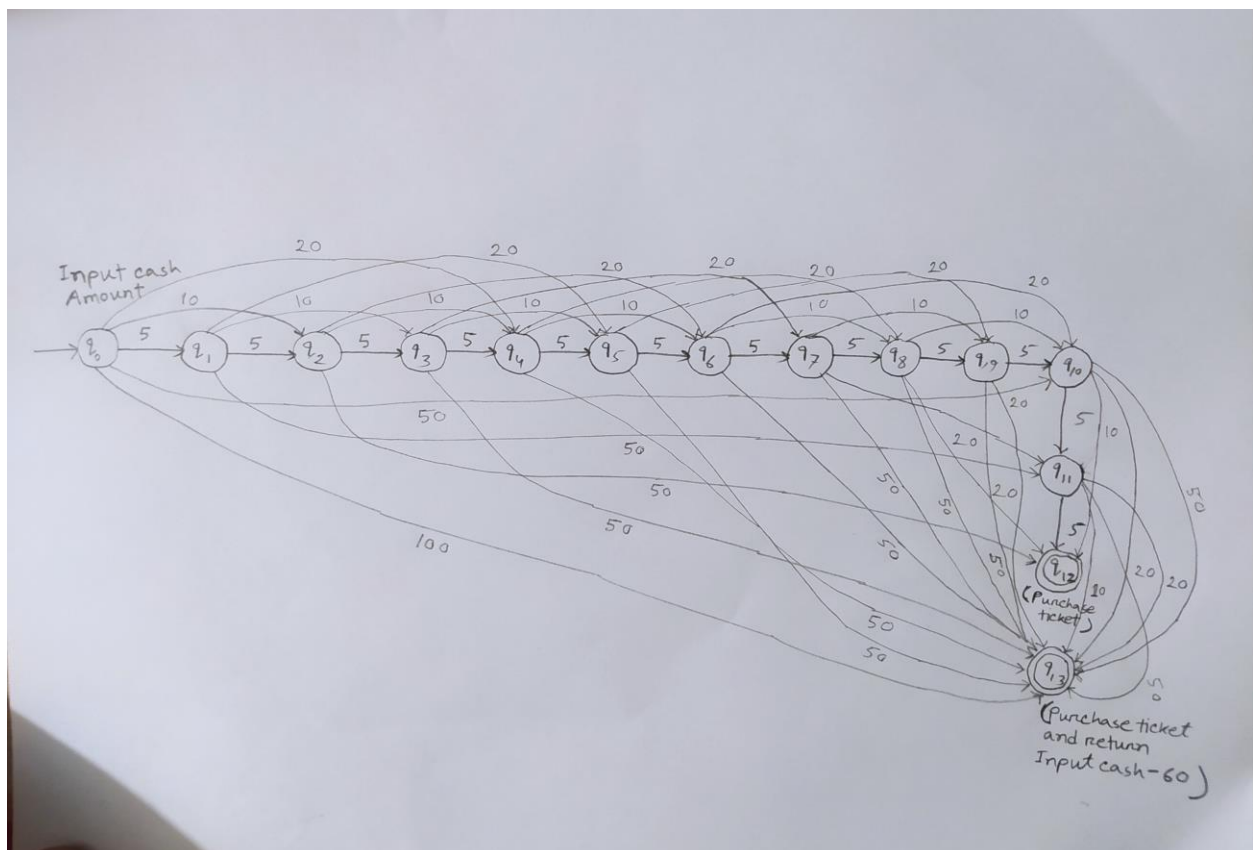


Figure 3.2: Design FSM of Vending Machine

Chapter IV.

4.1 Implementing the design through a programming language

The given problem is to design a ticket vending machine that accepts banknotes up to 100 Bangladeshi Taka and issues a ticket for a fare of 60 Taka. The machine should only accept banknotes that make up at least the fare amount and should return the remaining amount if the inserted banknotes exceed the fare amount.

To solve this problem, we can use a Finite State Machine (FSM) approach. The FSM is a mathematical model that describes the behavior of a system based on a finite number of states, inputs, and outputs. It is a useful tool for designing systems that have a finite number of states and can be represented using a state diagram. In the context of the ticket vending machine problem, we can represent the system using an FSM that has three types of states - Rejected, Accepted, and Returned. The Rejected state indicates that the machine has not received enough banknotes to issue a ticket. The Accepted state indicates that the machine has received the exact amount of banknotes to issue a ticket. The Returned state indicates that the machine has received more banknotes than the ticket fare and will return the remaining amount.

To implement the FSM in a programming language (structured or OOP based language), we can use conditional statements and loops to represent the different states and transitions. We can also use variables to store the amount of banknotes received and the remaining amount to be returned.

In the Rejected state, we can use a loop to accept further banknotes until the total amount received is equal to or greater than the fare amount. If the total amount is still less than the fare amount, we can display an appropriate message to the user that “Insufficient amount received. Please enter more banknotes”.

In the Accepted state, we can display a message to the user indicating that the ticket has been purchased and further banknotes will not be accepted.

In the Returned state, we can display a message to the user indicating that the ticket has been issued and the remaining amount will be returned. We can then use a loop to return the remaining amount by subtracting the fare amount from the total amount received and dispensing the banknotes until the remaining amount is zero.

Also depending on the programming methodology the same problem has different solution as well could be follow different approach. Sometimes different language C, C++, Java, JavaScript gives different approach solution by applying some method but implement the same problem with the same FSM logic.

Overall, the FSM approach provides a systematic way of designing and implementing the ticket vending machine problem. By using a programming language, we can easily translate the FSM model into executable code and test the system for different scenarios. implementing the ticket vending machine problem with programming provides a reliable, efficient, and convenient solution for handling ticket sales in various settings.

Chapter V.

5.1 Source code of implementing FSM

Program file name: minkhayer.c

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

int main() {
    int ticket_fare = 60;
    int total_value = 0;
    char input[100];

    printf("Enter banknotes to purchase the ticket:\n");
    fgets(input, 100, stdin);

    char *token = strtok(input, " ");
    while (token != NULL) {
        int banknote = atoi(token);

        if (banknote <= 0 || banknote > 100) {
            printf("Invalid banknote. Please enter a valid banknote between 1 and 100 Taka.\n");
        }
        else {
            total_value += banknote;

            if (total_value < ticket_fare) {
                printf("Total amount received so far: %d.\n", total_value);
            }
        }
    }
}
```

```

else if (total_value == ticket_fare) {
    printf("Total amount received: %d. Ticket purchased.\n", total_value);
    return 0;
}
else {
    int change = total_value - ticket_fare;
    printf("Total amount received: %d. Ticket purchased. Change: %d.\n", total_value,
change);
    return 0;
}
}

token = strtok(NULL, " ");
}

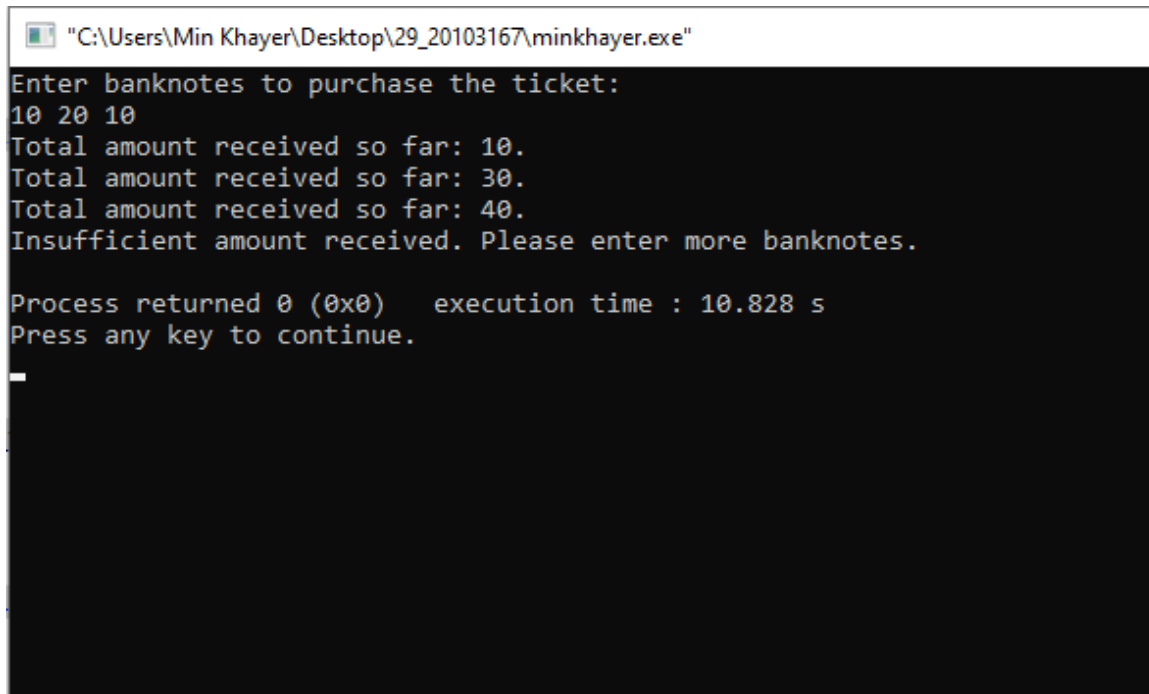
if (total_value < ticket_fare) {
    printf("Insufficient amount received. Please enter more banknotes.\n");
}

return 0;
}

```

Chapter VI.

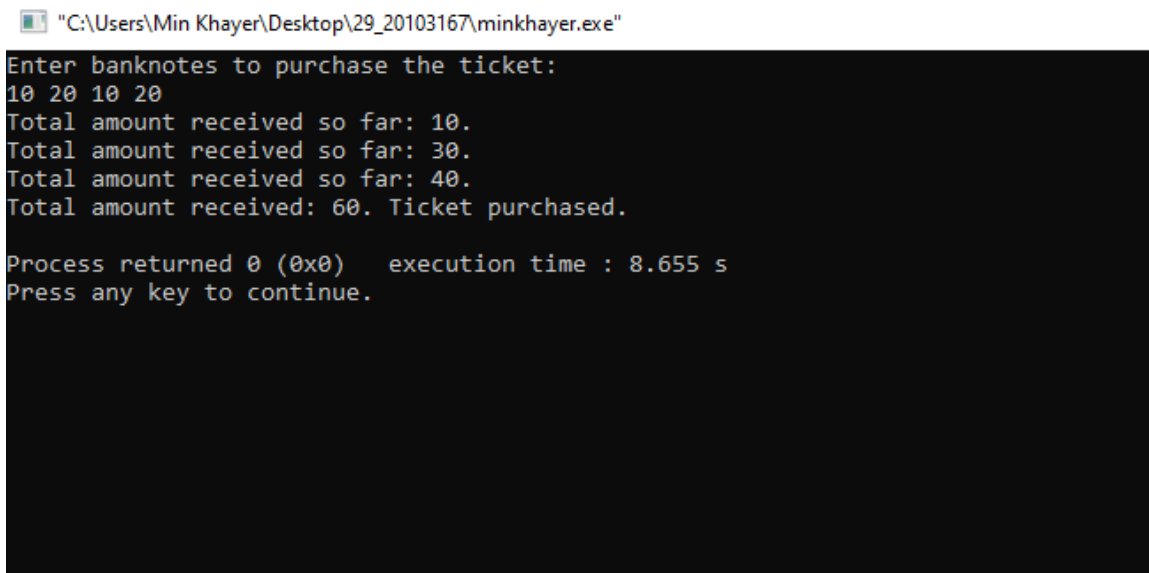
6.1 The output of the program of vending machine



```
"C:\Users\Min Khayer\Desktop\29_20103167\minkhayer.exe"
Enter banknotes to purchase the ticket:
10 20 10
Total amount received so far: 10.
Total amount received so far: 30.
Total amount received so far: 40.
Insufficient amount received. Please enter more banknotes.

Process returned 0 (0x0)   execution time : 10.828 s
Press any key to continue.
_
```

The output of this program here is the first scenario the input cash amount less than 60 taka that's why it is showing the message is insufficient amount received and please enter more banknotes.



```
"C:\Users\Min Khayer\Desktop\29_20103167\minkhayer.exe"
Enter banknotes to purchase the ticket:
10 20 10 20
Total amount received so far: 10.
Total amount received so far: 30.
Total amount received so far: 40.
Total amount received: 60. Ticket purchased.

Process returned 0 (0x0)   execution time : 8.655 s
Press any key to continue.
_
```

The output of this program here is the second scenario the input cash amount equal to 60 taka that's why it is showing the message is Ticket purchased.

```
"C:\Users\Min Khayer\Desktop\29_20103167\minkhayer.exe"
Enter banknotes to purchase the ticket:
20 50
Total amount received so far: 20.
Total amount received: 70. Ticket purchased. Change: 10.

Process returned 0 (0x0)   execution time : 8.649 s
Press any key to continue.
_
```

The output of this program here is the third scenario the input cash amount greater to 60 taka that's why it is showing the message is Ticket purchased and change the amount.