CIS-11 PROJECT DOCUMENTATION

TEAM MEMBERS:

AUDREY REINHARD

KATHYA JOSSELINE ROMANO TEPOZTECO

SERGIO ADAME

PROJECT NAME: BUBBLE SORT PROJECT

DATE: 05/19/2024

ADVISOR: KASEY NGUYEN, PhD

Objectives

- Know what the busiest hours of service are.
- Improve schedules and work time management.
- Increased teamwork among staff.
- Reduce accident expenses.
- Improve the quality of customer service.
- Get more customer approval.

Business Process

It is a coffee shop located near the University of California Riverside. It opened a few months ago and gained popularity in the first months of opening due to its space and because it is a good place to study for students since it offers free Wi-Fi access to its customers. The owners did not expect their business to succeed so soon. They noticed that, with the increase in customers, it was harder to offer quality service with the system with which they managed the working hours of their employees. The business has five part-time employees. Three of them work the first four hours, and two work the last four hours of the day. The workers on shift are in charge of taking orders, charging, making the product, and cleaning the facility. The owner has never adjusted hours since the opening of the business.

The company has problems with staff organization that affect production and customer service during busy hours since they are sometimes short on staff. This causes a decline in customer service because they are making the customers wait too long. There have also been monetary losses due to accidents caused by the stress which the staff is subjected to.

Owners need to know the busiest hours to predict and plan their workers' weekly schedules. Our idea is to implement a system that counts the number of sales during each hour of service. The system will then sort them in ascending order to help the business owner locate the busiest and most selling hours. The system will help the owner plan the shifts of their workers and offer them more working hours to deliver better customer service during those hours and minimize the potential for losses and accidents. As well as identifying the busiest hours will help the owner implement promotions to increase the popularity and sales of their product variety.

User Roles and Responsibilities.

Cashiers are in charge of taking and collecting orders from consumers, so they are an important part of the process. They enter each order, so the system can count the number of sales registered per hour and day. For this reason, and without exception, the cashier must enter each sale into the system.

The **manager** on duty is in charge of making the cash cutoff, which will allow the system to count the total sales by day and hour. Then, a daily copy with this information will be sent to the corresponding person to plan the schedules of the workers and another will be sent to the owner to make decisions about possible promotions or executive actions of the business.

The **owner** will receive a report on the number of sales and, according to that, a report on the possible working hours of each employee per week that will need to be approved before being implemented.

Production Rollout Considerations

In the implementation of this system, we will divide the staff into groups. One group will be in charge of the cleaning of the facility. A team will be exclusively responsible for the production of the products. Another team will be in charge of taking orders and the cash register. For this reason, they will receive the corresponding training to operate the system. The cashier must start the system at the beginning of the shift. The system will ask every hour for confirmation of the number of items sold per hour - we expect quantity from 0 to 100 items sold per hour. The user will enter this amount whenever the system asks for it. At the end of the day, when the cash cutoff is made, the manager will get a summary of the amounts entered during the day, and they will be organized in ascending order. The summary will be used in the weekly report that will be given to the owner.

Part II – Functional Requirements

Statement of Functionality:

This program takes eight inputs from the user (cashier), which are then compared and sorted from least to greatest. The system first compares two numbers to detect which number is greater than the other. After it finds which number is greater, it either swaps the two numbers or leaves them as is. The program will do this for all the numbers and then will do this process a few more times until all the numbers are sorted.

The program will use six registers. Registers help move and store information temporarily. In this case, the information that the cashier enters for each hour will be stored in registers to make the necessary calculations. It will store some of the information in places in memory that we call variables. To locate information in the computer's memory, we give an address for each variable. Thus, every time a piece of data is entered, the program will know where to locate it to perform its purpose.

To help us with data storage we will use an array. The list of numbers will be stored in a special type of variable called an array. An array holds multiple values and will make the process of looping through the values easier. There will be one array with a memory block for 9 spaces. The first spot (Array[0]) will hold the number of elements to be entered. The rest contain the numbers to be sorted. We will use a register as an index to calculate the addresses of each element.

There will be one loop to take the inputs of the integers. Then there will be two nested loops to sort through the array. The outer loop will be how many times the array is sorted. The inner loop will sort the two adjacent numbers through the array. Since we'll be working with eight different inputs, we will need a block of instruction to be repeated, so we will use a loop and subroutine to minimize the code. Within this block of instruction, arithmetic, and logical calculations will be made that will allow the program to work. Once the program has done the calculations, logic, and corresponding data sort, it will display the list of data in ascending order.

Scope

This small program can only be used by one user at a time. The program is designed for the cashier's use, and it will ask the user to input how many items the user sold per hour eight times, which is the number of service hours. The program does not ask about a specific product. It asks for the total quantity of all products sold in that hour. So, the number should be an integer between 1-100.

Performance

The program has an average performance since it uses the register R0-R7 and six variables. An overflow can be expected if the user enters any integer outside the 1-100 range because the program does not support integers outside that range.

Usability

The program is simple to use as it does not require extensive training. It requires a short class to teach the user to enter the data and the simple commands. The system asks a clear and specific question to the user: "Enter the total number of items sold during this hour (1-100)."

If the user needs to enter a number less than 1 or greater than 100, he must notify the manager as the system will show an error.

Documenting Requests for Enhancements

Date	**Enhancement**	**Requested by**	**Note**	**Priority**	**Release No/ Status**
05/16/2024	Get input from console	Audrey Reinhard	Get user input from console	Urgent	05/25/2024 (Pending)

05/16/2024	Handle triple digit inters.	Audrey Reinhard	Find a way to handle numbers above 99	Urgent	05/26/2024 (Pending)

Part III – Appendices/ Terminology

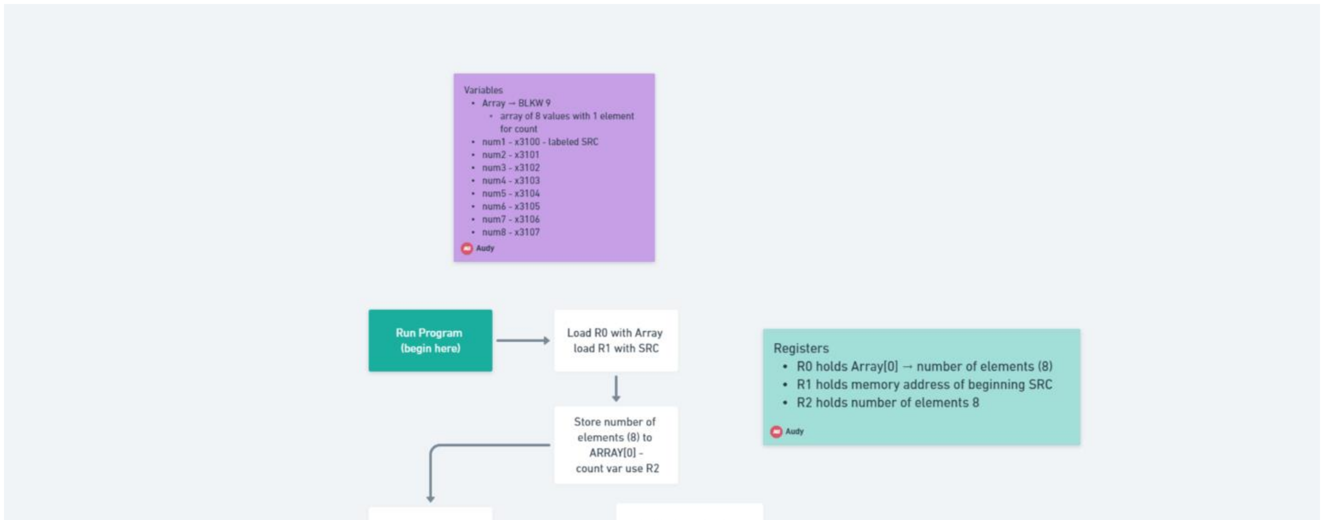
Memory address: It is a numerical value that contains the location in the computer's memory where data or instructions are stored.

Array: It is a sequence of locations in the computer's memory that allows data of the same type to be stored.

Loop: It is a block of code containing instructions that need to be repeated. A loop allows an efficient program as it saves time and lines of code.

Subroutine: It is a block of code called from multiple places in a program. A subroutine performs specific tasks that help reuse and organize code.

Flow chart or pseudo-code.


Variables
• Array → BLKW 9
• array of 8 values with 1 element for count
• num1 - x3100 - labeled SRC
• num2 - x3101
• num3 - x3102
• num4 - x3103
• num5 - x3104
• num6 - x3105
• num7 - x3106
• num8 - x3107
Audy
Run Program (begin here)
Load R0 with Array load R1 with SRC
Store number of elements (8) to ARRAY[0] - count var use R2
Registers
• R0 holds Array[0] → number of elements (8)
• R1 holds memory address of beginning SRC
• R2 holds number of elements 8
Audy