PRE-RELEASE MATERIAL May/June 2020

O Level Computer Science 2210/22

Abstract

This document provides solution to the Cambridge pre-release material for Computer Science 2210

/blitzcomputing

info@blitz.education

Prepared by: Blitz Computing

Pre-Release Material Tasks

A car park payment system allows customers to select the number of hours to leave their car in the car park. The customer will get a discount if they enter their frequent parking number correctly. The system calculates and displays the amount the customer must pay. The price of parking, the number of hours the customer can enter, and any discount depend upon the day of the week and the arrival time. The number of hours entered is a whole number. The price per hour is calculated using the price in force at the arrival time. No parking is allowed between Midnight and 08:00.

Day of the week	Arrival time			
	From 08:00 to 15:59		From 16:00 to Midnight	
	Max stay in hours	Price per hour	Hours	Price
Sunday	8	2.00	Up to Midnight	2.00
Monday	2	10.00	Up to Midnight	2.00
Tuesday	2	10.00	Up to Midnight	2.00
Wednesday	2	10.00	Up to Midnight	2.00
Thursday	2	10.00	Up to Midnight	2.00
Friday	2	10.00	Up to Midnight	2.00
Saturday	4	3.00	Up to Midnight	2.00

A frequent parking number can be entered for discounted parking. This number consists of 4 digits and a check digit that is calculated using a modulo 11 check digit calculation. A discount of 50% is available for arrival times from 16:00 to Midnight; the discount is 10% at all other arrival times.
Write and test a program or programs to simulate the car park payment system.

i. Your program or programs must include appropriate prompts for the entry of data; data must be validated on entry
ii. Error messages and other output need to be set out clearly and understandably.
iii. All variables, constants and other identifiers must have meaningful names

You will need to complete these **three** tasks. Each task must be fully tested

TASK 1 – Calculating the price to park

A customer inputs the day, the hour of arrival excluding minutes (for example 15:45 would be 15), the number of hours to leave their car, and a frequent parking number if available. If the frequent parking number has an incorrect check digit, then no discount can be applied. The price to park, based on the day, the hour of arrival, the number of hours of parking required and any discount available, is calculated and displayed.

TASK 2 – Keeping a total of the payments

Extend Task 1 to keep a daily total of payments made for parking. The daily total is zeroed at the start of the day. For the simulation, each customer inputs the amount paid, this must be greater than or equal to the amount displayed. There is no change given so the amount input may exceed the amount displayed.
Each customer payment is added to the daily total, and this total is displayed at the end of the day.

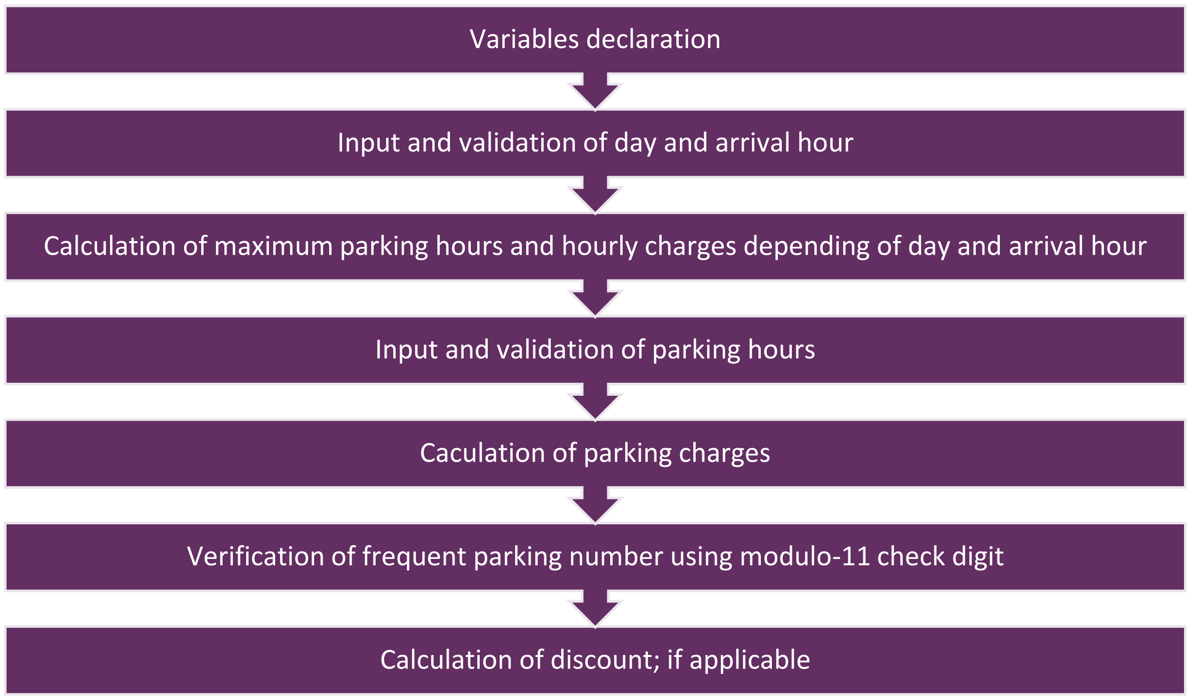
TASK 3 – Making payments fairer

Customers have complained that sometimes they are being charged too much if they arrive before 16:00 and depart after 16:00. Extend Task 1 to calculate the price before 16:00, then add the evening charge. For example, a customer arriving at 14:45 on a Sunday and parking for five hours was previously charged 10.00 and would now be charged 6.00

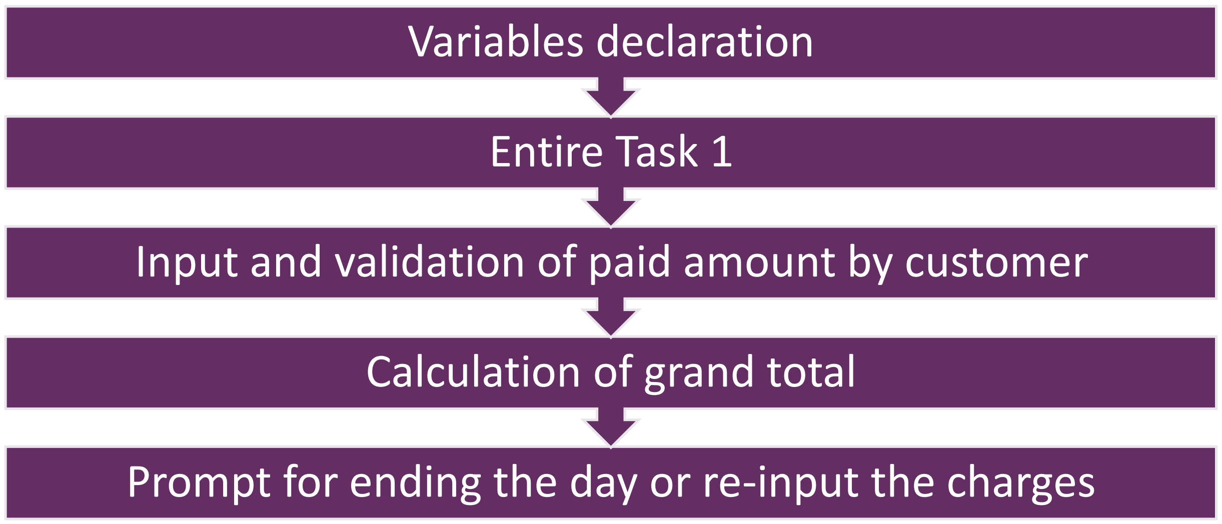
Concept and Understanding of Tasks

All 3 tasks are a part of one big problem i.e. setting up a parking system with different criteria for charge calculation. Both task 2 and task 3 modifies code/algorithm of task 1, hence all three tasks are considered separate programs. The general flow and explanation of each task is provided separately in the following diagram.

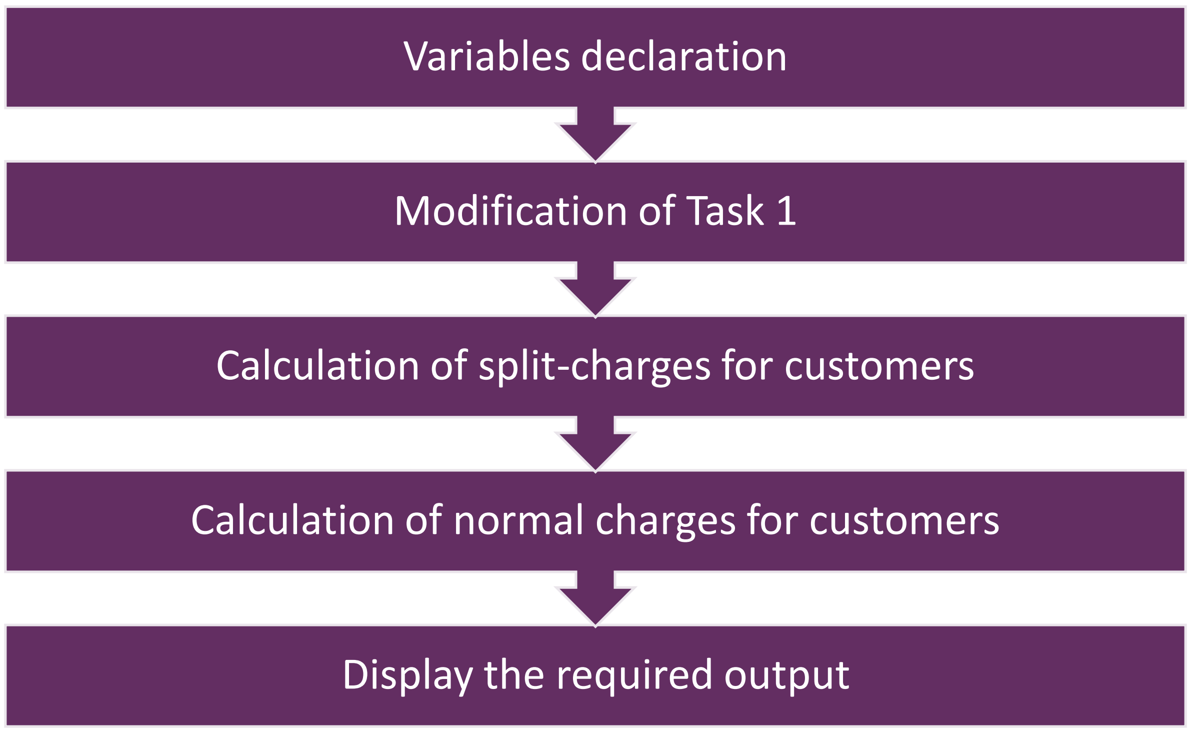
Task 1


Variables declaration
Input and validation of day and arrival hour
Calculation of maximum parking hours and hourly charges depending of day and arrival hour
Input and validation of parking hours
Caculation of parking charges
Verification of frequent parking number using modulo-11 check digit
Calculation of discount; if applicable

Task 2


Variables declaration
Entire Task 1
Input and validation of paid amount by customer
Calculation of grand total
Prompt for ending the day or re-input the charges

Task 3


Variables declaration
Modification of Task 1
Calculation of split-charges for customers
Calculation of normal charges for customers
Display the required output

Features of this Pre-Release

i. Complex selection (IF/CASE) statements
ii. Modulo-11 check digit calculation
iii. Overall Complexity: Easy / Intermediate / **Challenging**

Approach to Solution

Like every algorithm, there can be many possible approaches to solve these tasks depending upon the understanding of person. We are listing down the key points that reflect our understanding and we'll solve these tasks according to following **assumptions**.

1. In Task1, the check digit calculation needs to be done for verifying "frequent parking number". There are **two possible ways** to do this calculation and we will do it using a simplified method.

Color Codes

The pseudocode uses different colors to represent keywords for easier understanding. These color codes are listed below.

Begin / End	**BLACK**
Variable declaration and datatypes	**LIGHT BLUE**
Selection statements (IF and CASE)	**RED**
Input and Output	**GREEN**
Loop (REPEAT-UNTIL/WHILE)	**PURPLE**
Prompts, messages and variables	**BLACK**
Loop (FOR-NEXT)	**PINK**

Explanation of Algorithm of Tasks

The explanation of the algorithms used in each task is listed below.

Task 1

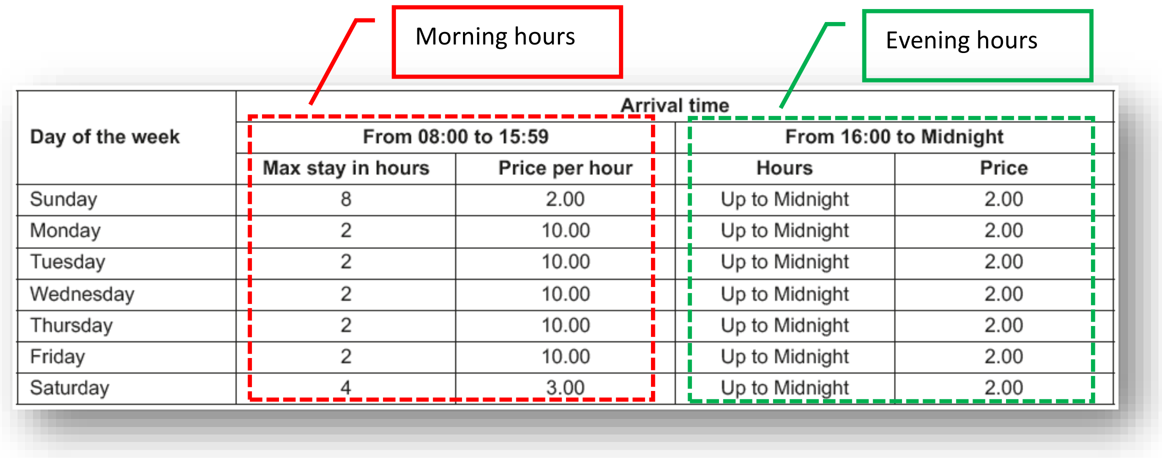
In this task we have to calculate the charges for parking the vehicle. If you read the pre-release completely, then you will quickly realize that a lot of information needs to be collected before you can calculate the parking charges and the information is scattered in bits and pieces.

Let's summarize everything that we need to do in Task 1:

1. Input the day of the week.
2. Input the arrival time.
3. Input the number of hours to park.
4. Calculate the parking charges depending upon the day (1) and arrival time (2). Also, take into consideration that charges are different depending upon the arrival time (2).
5. Input the "frequent parking number", if available, and verify it using modulo 11 check digit calculation.
    i. If "frequent parking number" is correct then apply discount depending upon the arrival time (2). For morning (8hrs – 15hrs) its 10% and for evening (16hrs – 24hrs) its 50%
6. Finally print the parking fee, hour of the arrival, number of parking hours and discount (if applicable)

All the inputs will be **validated using WHILE** statement which forces the user to input only valid data.

In order to calculate the parking charges, the person will first input the day and his arrival time. The price per hour and the maximum parking stay is further divided into two categories (**morning hours or evening hours**) depending upon the arrival time. If you notice, then morning timings for Sunday and Saturday have different charges and stay hours while all other days are same. Similarly evening charges and stay hours are **same for all days**, irrespective of the day.

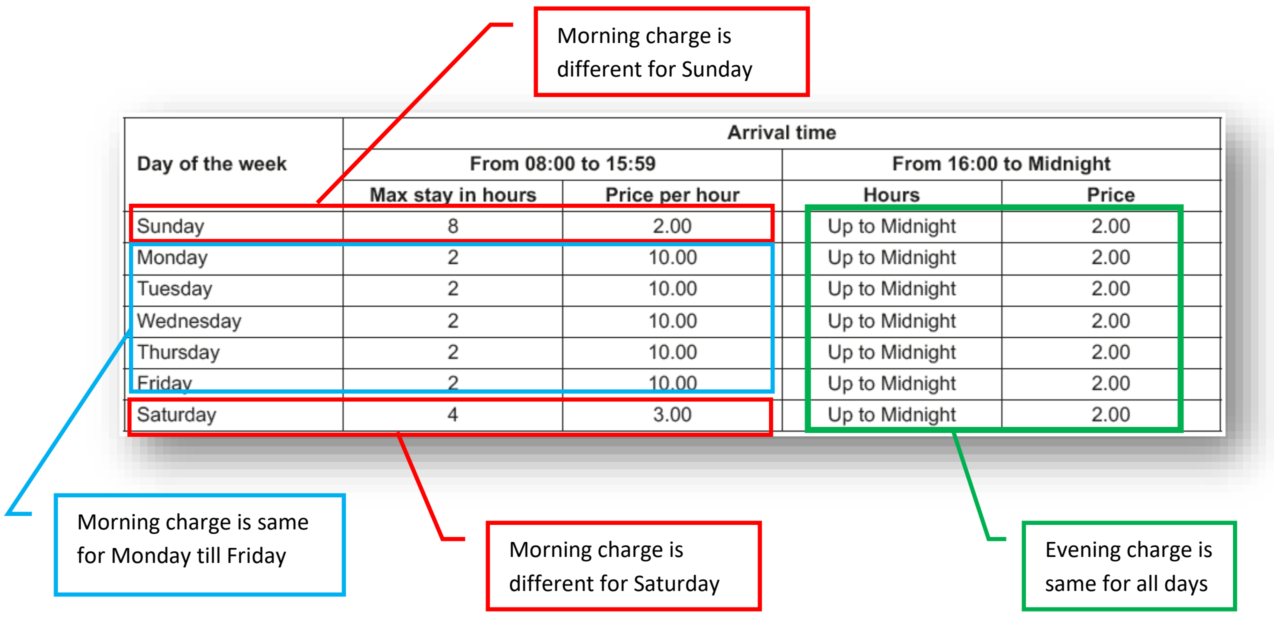

Morning hours
Evening hours

Day of the week	Arrival time			
	From 08:00 to 15:59		From 16:00 to Midnight	
	Max stay in hours	Price per hour	Hours	Price
Sunday	8	2.00	Up to Midnight	2.00
Monday	2	10.00	Up to Midnight	2.00
Tuesday	2	10.00	Up to Midnight	2.00
Wednesday	2	10.00	Up to Midnight	2.00
Thursday	2	10.00	Up to Midnight	2.00
Friday	2	10.00	Up to Midnight	2.00
Saturday	4	3.00	Up to Midnight	2.00

So, there are 4 possibilities which decide the charges of parking. These are:

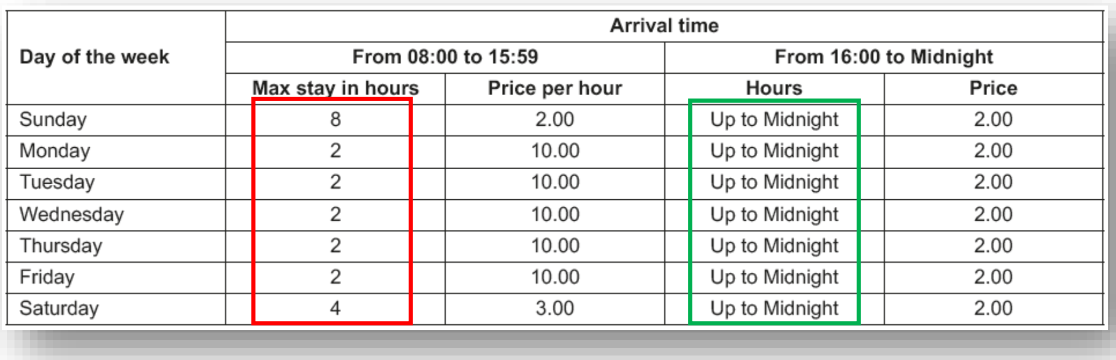
1) Person comes on Sunday and between 8 to 15hrs.
2) Person comes on Saturday and between 8 to 15hrs
3) Person comes on any day from Monday to Friday and between 8 to 15hrs
4) Person comes on any day after 15hrs

All of the above conditions will be implemented using IF statements in our program.


Morning charge is different for Sunday
Morning charge is same for Monday till Friday
Morning charge is different for Saturday
Evening charge is same for all days

Day of the week	Arrival time			
	From 08:00 to 15:59		From 16:00 to Midnight	
	Max stay in hours	Price per hour	Hours	Price
Sunday	8	2.00	Up to Midnight	2.00
Monday	2	10.00	Up to Midnight	2.00
Tuesday	2	10.00	Up to Midnight	2.00
Wednesday	2	10.00	Up to Midnight	2.00
Thursday	2	10.00	Up to Midnight	2.00
Friday	2	10.00	Up to Midnight	2.00
Saturday	4	3.00	Up to Midnight	2.00

Now we will input the number of hours for parking stay and validate it according to the given criteria (highlighted in table below) which allows us to calculate the parking charges.



Day of the week	Arrival time			
	From 08:00 to 15:59		From 16:00 to Midnight	
	Max stay in hours	Price per hour	Hours	Price
Sunday	8	2.00	Up to Midnight	2.00
Monday	2	10.00	Up to Midnight	2.00
Tuesday	2	10.00	Up to Midnight	2.00
Wednesday	2	10.00	Up to Midnight	2.00
Thursday	2	10.00	Up to Midnight	2.00
Friday	2	10.00	Up to Midnight	2.00
Saturday	4	3.00	Up to Midnight	2.00

At this stage we have all the relevant data needed to calculate the parking charges. We will now move on to **apply discount** which depends upon a "frequent parking number".

According to the pre-release, the verification of "frequent parking number" needs to be done using modulo 11 check digit method and if it found to be correct, then apply the discount percentage (which is 10% of the total charges in morning timings and 50% on evening timings). It is mentioned that the "frequent parking number" is a 4-digit number plus a check digit i.e. total 5-digits number.

The modulo-11 check digit verification is quite simple and explained below.

1. Each digit is assigned a **weight** from 1 to 5, starting from right-most digit.
2. The weights are **multiplied** with their respective digits.
3. The values obtained (from step 2) are **added** together.
4. The result of step 3 is **divided (MOD) by 11** and its remainder is recorded.
5. If the **remainder is zero** then the **number is correct**, otherwise it contains some error.

Example:

Let's verify the frequent parking number 43573 and see if it valid.

Weight	5	4	3	2	1
Parking number	**4**	**3**	**5**	**7**	**3**
Step 2 (multiplying by weight)	20	12	15	14	3
Step 3 (adding values)	20 + 12 + 15 + 14 + 3				
Step 4 (Dividing by 11 to get remainder)	64 MOD 11 = 9				
Result	**Invalid**				

Now there are two ways through which we can implement the above logic using pseudocode; ***loop based calculation*** and ***individual digit based calculation***, both are listed below:

Loop-based calculation code:

```
INPUT "Enter frequent parking number", parknum
FOR count ← 1 TO 5
    digit ← parknum MOD 10
    sum ← sum + (digit * count)
    parknum ← (parknum – digit) / 10
NEXT
remainder ← sum MOD 11
IF remainder = 0 THEN
    PRINT "Valid parking number"
ELSE
    PRINT "Invalid parking number"
END IF
```

Individual digit-based calculation code:

```
INPUT "Enter frequent parking number", d1, d2, d3, d4, checkdigit
sum ← (d1 * 5) + (d2 * 4) + (d3 * 3) + (d4 * 2) + (checkdigit * 1)
remainder ← sum MOD 11
IF remainder = 0 THEN
    PRINT "Valid parking number"
ELSE
    PRINT "Invalid parking number"
END IF
```

In "individual digit" method, we are simply taking input of each digit separately in different variables (d1, d2, d3, d4, checkdigit). This makes this method easier to understand as compare to the loop based method and we will be utilizing it in our pseudocode.

There is one potential problem that needs to be taken care of in the code. Since we are using modulo 11 check digit method so there is a possibility that the check digit may be 10 which is usually represented by X. We need to make sure (by displaying a message) that the user enters this X as 10 in the program otherwise the program will not work correctly)

We will now apply the discount to the total parking charges if the "frequent parking number" is valid and print all the required information i.e. total price/charges, hour of arrival, number of parking hours and discount.

Task 2

In this task, we will calculate the grand total of all parking charges collected during the day. The task is extremely simple and we can easily setup the REPEAT loop to enclose entire task 1 and add code in the end to input the payment and add it to the grand total. The only validation we need to perform is to make sure that payment entered by the user should be equal or greater than the calculated charges. We will then present the user an option (using REPEAT loop) to end the day so grand total can be displayed.

Task 3

In this task we need to improve the parking charge calculation done in task 1. If you noticed in task 1, then there is a fundamental flaw in the charge calculation. The flaw impacts those persons who park their cars before 16hrs and depart after 16hrs.
For example, if a person arrival time is 15hrs on Saturday, then he is allowed the maximum parking time of 4 hours. Let's say he park the car for 4 hours so his departure time would be 19hrs and his total charges would be 4hrs x $3 = $12. But according to the given chart of price, the parking charges reduces to flat-rate of $2 after 16hrs time mark. So the person should really be paying (1hr x $3) + $2 = $5, but our calculation didn't take this into consideration and apply the charges of morning hours even when the charges should be changed to evening and only $2 should be **added**.

Task 1 Solution (Pseudocode)

```
BEGIN
    DECLARE day ← 0, hourarrival ← 0, noofhours ← 0, parkingnum ← 0, maxhours ← 0
                remainder ← 0 AS INETGER
    DECLARE d1 ← 0, d2 ← 0, d3 ← 0, d4 ← 0, checkdigit ← 0 AS INTEGER
    DECLARE discount ← 0.0, discountamount ← 0.0, hourlyprice ← 0.0, totalprice ← 0.0 AS FLOAT
    DECLARE choice AS CHAR

    INPUT “Enter day. (Sunday=1, Monday=2 till Saturday=7)”, day
    WHILE day < 1 OR day > 1
        INPUT “Wrong input. Day should be between 1 to 7”, day
    END WHILE

    INPUT “Enter arrival hour. (8 to 23)”, hourarrival
    WHILE hourarrival < 8 OR hourarrival > 23
        INPUT “Wrong input. Arrival hour should be between 8 and 23”, arrivalhour
    END WHILE

    IF day = 1 AND arrivalhour < 16 THEN
        maxhours ← 8
        hourlyprice ← 2.00
    ELSE IF day = 7 AND arrivalhour < 16 THEN
        maxhours ← 4
        hourlyprice ← 3.00
    ELSE IF arrivalhour < 16 THEN
        maxhours ← 2
        hourlyprice ← 10.00
    ELSE
        maxhours ← 24 - hourarrival
        hourlyprice ← 2.00
    END IF

    PRINT “Allowed max parking hours are “, maxhours
    INPUT “Enter number of hours to park the vehicle”, noofhours
    WHILE noofhours < 1 OR noofhours > maxhours
        PRINT “Wrong input. Max hours allowed are ”, maxhours
        INPUT noofhours
    END WHILE
    IF hourarrival < 16 THEN
        totalprice ← noofhours * hourlyprice
        discount ← 0.1
    ELSE
        totalprice ← hourlyprice
        discount ← 0.5
    END IF

    INPUT "Do you want to enter frequent parking number? (Y/N)", choice
    IF choice = 'Y' THEN
        INPUT "Enter frequent parking number. Please note if your number's last digit is X, then
                    you have to enter 10 in its place" , d1, d2, d3, d4, checkdigit
        remainder ← (d1 * 5) + (d2 * 4) + (d3 * 3) + (d4 * 2) + (checkdigit * 1)
        remainder ← remainder MOD 11
        IF remainder ← 0 THEN
            discountamount ← totalprice * discount
            totalprice ← totalprice – discountamount
        END IF
    END IF

    PRINT "The arrival hour is ", arrivalhour
    PRINT "Number of hours of parking are ", noofhours
    PRINT "Discount applied is ", discountprice
    PRINT "Total payment of parking is ", totalprice

END
```

Efficiency of Algorithm

- All identifiers are initialized with default values.
- Use of WHILE loop to validate all user inputs
- Verification of parking number using modulo 11 check digit

Pseudocode Explanation of Task 1

```
BEGIN
    DECLARE day ← 0, hourarrival ← 0, noofhours ← 0, parkingnum ← 0, maxhours ← 0
            remainder ← 0 AS INETGER
    DECLARE d1 ← 0, d2 ← 0, d3 ← 0, d4 ← 0, checkdigit ← 0 AS INTEGER
    DECLARE discount ← 0.0, discountamount ← 0.0, hourlyprice ← 0.0, totalprice ← 0.0 AS FLOAT
    DECLARE choice AS CHAR
```

Declaration of variables

```
    INPUT "Enter day. (Sunday=1, Monday=2 till Saturday=7)", day
    WHILE day < 1 OR day > 1
        INPUT "Wrong input. Day should be between 1 to 7", day
    END WHILE
```

Input and validation of day using WHILE statement. The day will be entered as numbers where Sunday=1, Monday=2 till Saturday=7

```
    INPUT "Enter arrival hour. (8 to 23)", hourarrival
    WHILE hourarrival < 8 OR hourarrival > 23
        INPUT "Wrong input. Arrival hour should be between 8 and 23", arrivalhour
    END WHILE
```

Input and validation of arrival hour. It should be between 8hrs and 23hrs. It cannot be 24hrs as parking is not allowed "from" midnight i.e. 24hrs

```
    IF day = 1 AND arrivalhour < 16 THEN
        maxhours ← 8
        hourlyprice ← 2.00
    ELSE IF day = 7 AND arrivalhour < 16 THEN
        maxhours ← 4
        hourlyprice ← 3.00
    ELSE IF arrivalhour < 16 THEN
        maxhours ← 2
        hourlyprice ← 10.00
    ELSE
        maxhours ← 24 - hourarrival
        hourlyprice ← 2.00
    END IF
```

The primary statements which will set the values of hourly price and max parking hours allowed. We will use these values later to calculate the total price. As explained earlier, there can be only 4 ways in which the price can be calculated

Sunday and morning (8 to 15)

Saturday and morning (8 to 15)

All other days and morning (8 to 15)

All days and evening (16 to 24)

```
    PRINT "Allowed max parking hours are ", maxhours
    INPUT "Enter number of hours to park the vehicle", noofhours
    WHILE noofhours < 1 OR noofhours > maxhours
        PRINT "Wrong input. Max hours allowed are ", maxhours
        INPUT noofhours
    END WHILE
```

Input of max hours of parking. The hours allowed has been calculated in the above IF statements so validation can be easily performed.

```
IF hourarrival < 16 THEN
    totalprice ← noofhours * hourlyprice
    discount ← 0.1
ELSE
    totalprice ← hourlyprice
    discount ← 0.5
END IF
```

Calculation of parking charges. The IF statement calculates the charges depending upon arrival time of morning (8-15) or evening (16-23). Also note that discount variable has been given the value here so it can be used later in the code.

```
INPUT "Do you want to enter frequent parking number? (Y/N)", choice
IF choice = 'Y' THEN
    INPUT "Enter frequent parking number. Please note if your number's last digit is X, then
                you have to enter 10 in its place" , d1, d2, d3, d4, checkdigit
    remainder ← (d1 * 5) + (d2 * 4) + (d3 * 3) + (d4 * 2) + (checkdigit * 1)
    remainder ← remainder MOD 11
    IF remainder ← 0 THEN
        discountamount ← totalprice * discount
        totalprice ← totalprice – discountamount
    END IF
END IF
```

Verification of frequent parking number using check digit. The discount is only applied if the check digit calculation becomes successful.

```
PRINT "The arrival hour is ", arrivalhour
PRINT "Number of hours of parking are ", noofhours
PRINT "Discount applied is ", discountprice
PRINT "Total payment of parking is ", totalprice
END
```

Required output of Task 1

Task 2 Solution (Pseudocode)

** Since this task extends task 1, only newly added variables & code is shown. The location of task 1 code is mentioned clearly.*

```
BEGIN
    [ALL IDENTIFIERS OF TASK 1]
    DECLARE daytotal ← 0.0, paidamount ← 0.0 AS FLOAT
    DECLARE dayrepeat AS CHAR

    REPEAT
        [ENTIRE TASK 1 CODE]

        INPUT “Enter the amount to pay”, paidamount
        WHILE paidamount < totalprice
            INPUT “Wrong input. Amount should be equal or greater than total”, paidamount
        END WHILE
        daytotal ← daytotal + paidamount

        INPUT “Do you want to end the day and display the earnings? (Y/N)”, dayrepeat
    UNTIL dayrepeat = ‘Y’

    PRINT “Total earning for today is”, daytotal
END
```

Pseudocode Explanation of Task 2

```
BEGIN
    [ALL IDENTIFIERS OF TASK 1]
    DECLARE daytotal ← 0.0, paidamount ← 0.0 AS FLOAT
    DECLARE dayrepeat AS CHAR
    REPEAT
        [ENTIRE TASK 1 CODE]

        INPUT "Enter the amount to pay", paidamount
        WHILE paidamount < totalprice
            INPUT "Wrong input. Amount should be equal or greater than total", paidamount
        END WHILE
        daytotal ← daytotal + paidamount

        INPUT "Do you want to end the day and display the earnings? (Y/N)", dayrepeat
    UNTIL dayrepeat = 'Y'
    PRINT "Total earning for today is", daytotal
END
```

Declaration of variables of Task1 and Task 2.

Calculation of paid amount by the customer. The amount should be either equal or greater than the total.

Running sum of paid amount

If the user enters 'N', then whole program will run again otherwise total of the day will be printed

Output of earnings of the day

Task 3 Solution (Pseudocode)

** Since this task modifies task 1 code, it is rewritten completely.*

```
BEGIN
        DECLARE day ←0, hourarrival ←0, noofhours ←0, parkingnum ←0, maxhours ←0
                remainder ←0, hourdepart ←0 AS INETGER
    DECLARE d1 ←0, d2 ←0, d3 ←0, d4 ←0, checkdigit ←0 AS INTEGER
    DECLARE discount ←0.0, discountamount ←0.0, hourlyprice ←0.0, totalprice ←0.0 AS FLOAT
    DECLARE choice AS CHAR

    INPUT "Enter day. (Sunday=1, Monday=2,- Saturday=7)", day
    WHILE day < 1 OR day > 1
        INPUT "Wrong input. Day should be between 1 to 7", day
    END WHILE

    INPUT "Enter arrival hour. (8 to 23)", hourarrival
    WHILE hourarrival < 8 OR hourarrival > 23
        INPUT "Wrong input. Arrival hour should be between 8 and 23", arrivalhour
    END WHILE

    IF day = 1 AND arrivalhour < 16 THEN
        maxhours ←8
        hourlyprice ←2.00
    ELSE IF day = 7 AND arrivalhour < 16 THEN
        maxhours ←4
        hourlyprice ←3.00
    ELSE IF arrivalhour < 16 THEN
        maxhours ←2
        hourlyprice ←10.00
    ELSE
        maxhours ← 24 - hourarrival
        hourlyprice ←2.00
    END IF

    PRINT "Allowed max parking hours are ", maxhours
    INPUT "Enter number of hours to park the vehicle", noofhours
    WHILE noofhours < 1 OR noofhours > maxhours
        PRINT "Wrong input. Max hours allowed are: ", maxhours
        INPUT noofhours
    END WHILE
    hourdepart = hourarrival + noofhours
    IF hourarrival < 16 AND hourdepart > 16 THEN
        noofhours ← noofhours – (hourdepart - 16)
        totalprice ← (noofhours * hourlyprice) + 2
        discount ← 0.1
    ELSE IF hourarrival < 16 AND hourdepart <= 16 THEN
        totalprice ← noofhours * hourlyprice
        discount ← 0.1
    ELSE
        totalprice ← hourlyprice
        discount ← 0.5
    END IF

    INPUT “Do you want to enter frequent parking number? (Y/N)”, choice
    IF choice = ‘Y’ THEN
        INPUT “Enter frequent parking number. Please note if your number’s last digit is X, then
                you have to enter 10 in its place” , d1, d2, d3, d4, checkdigit
        remainder ← (d1 * 5) + (d2 * 4) + (d3 * 3) + (d4 * 2) + (checkdigit * 1)
        remainder ← remainder MOD 11
        IF remainder ← 0 THEN
            discountamount ← totalprice * discount
            totalprice ← totalprice – discountamount
        END IF
    END IF

    PRINT “The arrival hour is ”, arrivalhour
    PRINT “Number of hours of parking are ”, noofhours
    PRINT “Discount applied is ”, discountprice
    PRINT “Total payment of parking is ”, totalprice

END
```

Pseudocode Explanation of Task 3

** Only the parking charges part is explained as the other parts of the code as same as task 1*

```
hourdepart = hourarrival + noofhours
IF hourarrival < 16 AND hourdepart > 16 THEN
    noofhours ← noofhours – (hourdepart - 16)
    totalprice ← (noofhours * hourlyprice) + 2
    discount ← 0.1
ELSE IF hourarrival < 16 AND hourdepart <= 16 THEN
    totalprice ← noofhours * hourlyprice
    discount ← 0.1
ELSE
    totalprice ← hourlyprice
    discount ← 0.5
END IF
```

Calculation of departure hour so we can identify if the person comes before 16 and depart after 16

Calculation of price for all three scenarios. (mentioned below)

Split calculation for person arriving before 16hrs and depart after 16hrs

Calculation for person arriving and depart before 16hrs

Calculation for person arriving at or after 16hrs

Practice Questions

1. When you performed the tasks, you used variables. Write suitable declarations for two of these. State what you used each one for.
2. Write an algorithm to complete Task 1, using either pseudocode, programming statements or a flowchart.
3. Write an algorithm to complete Task 2, using either pseudocode, programming statements or a flowchart.
4. Write an algorithm to show how you corrected the price calculation in Task 3. Use either pseudocode, programming statements or a flowchart.
5. Explain how you performed validation check for any two inputs in Task 2. You can include pseudocode or programming statements as part of your explanation.
6. Explain how your program identify incorrect “parking numbers” entered by the user. You can include pseudocode or programming statements as part of your explanation.
7. Write an algorithm in pseudocode or program code to show the modification needed to give back change to the user if he enters amount greater than the parking charges.
8. The parking manager decides to change the criteria of discount given to the customers. Now the customers would be given 2% discount if they park the car for more than 3 hours on any day and any time. This discount is only for the customers who doesn’t have frequent parking number. Modify and rewrite your code of task 1 to accommodate this change.
9. Comment on the efficiency of your design for Task 1.

Comments & Feedback

This document is prepared by **Blitz Computing** to help students prepare for Computer Science 2210 Paper 2 as well as for teachers delivering the same to their students.

We would like to hear your **comments, feedback and suggestions** which will **motivate** us providing quality content to students and teachers. If you find any mistake, feel free to inform us!

Visit our Facebook page at: www.facebook.com/blitzcomputing
OR
Email us at: info@blitz.education

Thank You!

Blitz Computing Team.