Your preparation for the examination should include attempting the following practical tasks by **writing and testing a program or programs**.

An electric mountain railway makes four return trips every day. In each trip the train goes up the mountain and back down. The train leaves from the foot of the mountain at 09:00, 11:00, 13:00 and 15:00. The train returns from the top of the mountain at 10:00, 12:00, 14:00 and 16:00. Each train has six coaches with eighty seats available in each coach. Passengers can only purchase a return ticket; all tickets must be purchased on the day of travel. The cost is $25 for the journey up and $25 for the journey down. Groups of between ten and eighty passengers inclusive get a free ticket for every tenth passenger, provided they all travel together (every tenth passenger travels free). Passengers must book their return train journey, as well as the departure train journey, when they purchase their ticket. Passengers can return on the next train down the mountain or a later train. The last train from the top of the mountain has two extra coaches on it.

The train times are displayed on a large screen, together with the number of tickets still available for each train. Every time a ticket is booked the display is updated. When a train is full, the word 'Closed' is displayed instead of the number of tickets available.

Write and test a program or programs for the electric mountain railway.

- Your program or programs must include appropriate prompts for the entry of data; data must be validated on entry.
- Error messages and other output need to be set out clearly and understandably.
- 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 – Start of the day.
Write a program to set up the screen display for the start of the day. Initialise suitable data structure(s) to total passengers for each train journey and total the money taken for each train journey. Each train journey must be totalled separately. There are four journeys up and four journeys down every day.

Task 2 – Purchasing tickets.
Tickets can be purchased for a single passenger or a group. When making a purchase, check that the number of tickets for the required train journeys up and down the mountain is available. If the tickets are available, calculate the total price including any group discount. Update the screen display and the data for the totals.

Task 3 – End of the day.
Display the number of passengers that travelled on each train journey and the total money taken for each train journey. Calculate and display the total number of passengers and the total amount of money taken for the day. Find and display the train journey with the most passengers that day.

Task 1 – Start of the day.
Write a program to set up the screen display for the start of the day. Initialise suitable data structure(s) to total passengers for each train journey and total the money taken for each train journey. Each train journey must be totalled separately. There are four journeys up and four journeys down every day.

```
//Initialization of suitable data structures (arrays)

INTEGER Time[1:8] ← {9,11, 13, 15, 10, 12, 14, 16}   //Time array indices 1 to 4 are for Bottom to top train
                                                     //Time array indices 5 to 8 are for Top to bottom train

INTEGER Tickets[1:8] ← {480, 480, 480, 480, 480, 480, 480, 640 }
                                                     //Tickets array indices 1 to 4 are for Bottom to top train
                                                     //Tickets array indices 5 to 8 are for Top to bottom train

INTEGER FreeTickets[1:8] ←{0, 0, 0, 0, 0, 0, 0, 0}   // FreeTickets array indices 1 to 4 are for Bottom to top train
                                                     // FreeTickets array indices 5 to 8 are for Top to bottom train


//Variable declaration and intialization

INTEGER     Count ←0,     Passengers ←0

REAL        AmountPerTrain ← 0.0          // AmountPerTrain: To store amount for each single train


//CONSTANT declaration
INTEGER  CONST  GroupMin  ←  10          //Minimum members in a group (used in Task 2)
INTEGER  CONST  GroupMax  ←  80          //Maximum members in a group (used in Task 2)
INTEGER  CONST  TotalTrains ←  8         //Up + Down trains (used in Task 3)


REAL  CONST  RoundTripTicketCost ← 50.0  //Round trip ticket cost 25.0 + 25.0 =50.0 (used in Task 2)
REAL  CONST  OneWayTicketCost    ← 25.0  //One-way ticket cost 25.0 (used in Task 3)


                                         //Screen display for the start of the day

FOR Count ← 1 TO 4
    OUTPUT "Departure time is ", Time[Count], ":00 and", Tickets[Count], " ticket(s) still available in this train"

    OUTPUT "Return time is ", Time[Count + 4], ":00 and", Tickets[Count + 4], " ticket(s) still available in this train"
NEXT Count

                                         // [Count + 4] to retrieve the return (top to bottom) trains data
```

TASK 1 OUTPUT:

```
Departure time is 9:00 and 480 ticket(s) still available in this train
Return time is 10:00 and 480 ticket(s) still available in this train
Departure time is 11:00 and 480 ticket(s) still available in this train
Return time is 12:00 and 480 ticket(s) still available in this train
Departure time is 13:00 and 480 ticket(s) still available in this train
Return time is 14:00 and 480 ticket(s) still available in this train
Departure time is 15:00 and 480 ticket(s) still available in this train
Return time is 16:00 and 640 ticket(s) still available in this train
```


//Note:
//Double slash is used to write **single-line** comments.
//A **comment** is an explanation or description of the source code of the program. It helps a developer to explain the
//logic of the code and improves program readability. At run-time, a comment is ignored by the compiler.

Task 2 – Purchasing tickets.
Tickets can be purchased for a single passenger or a group. When making a purchase, check that the number of tickets for the required train journeys up and down the mountain is available. If the tickets are available, calculate the total price including any group discount. Update the screen display and the data for the totals.

```
//Task 2 Variable declaration and intialization
CHAR      Choice ← 'N'
INTEGER   CountOfTens ← 0,   TicketsPurchased ← 0,   DepartTrain ← 0,   ReturnTrain ←0
REAL      Price ← 0.0
BOOLEAN   Flag ← FALSE
STRING    Password ← " "                    // " ": Empty string
```

Flag variable is used as a signal in programming to let the program know that a certain condition has met.

REPEAT

```
                                        //Hour(Now): Returns current system hour (integer 0 through 23).
FOR Count ← 1 TO 4
    IF ( Tickets[Count] > 0 AND Time[Count] >  Hour(Now) )THEN
        OUTPUT "Departure time is ", Time[Count], ":00 and", Tickets[Count], " ticket(s) still available in this train"
    ELSE
        OUTPUT "Train at departure time ", Time[Count], ":00 is CLOSED"
    ENDIF
                                        // (Hour(Now) + 1): For example, if you didn't
                                        //depart at 9:00 you couldn't return at 10:00
    IF (Tickets[Count + 4] > 0 AND Time[Count + 4] >  (Hour(Now) + 1) ) THEN
        OUTPUT "Return time is ", Time[Count + 4], ":00 and", Tickets[Count + 4], " ticket(s) still available in this train"
    ELSE
        OUTPUT "Train at return time ", Time[Count + 4], ":00 is CLOSED"
    ENDIF
NEXT Count
```

```
IF ( Hour(Now) >= 0 AND Hour(Now) < 15) THEN        //Last train depart at exact 15:00, so tickets could be
                                                    //purchased only before 15:00.
{
        REPEAT                                      //Hour(Now): Returns current system hour (integer 0 through 23)
                OUTPUT "Dear traveller, do you want to book your journey? (Y/N)"
                INPUT Choice                                        //Character Check
        UNTIL (Choice = 'Y' OR Choice = 'N')
}
ELSE
{
        OUTPUT "No more trains available today. Please come tomorrow"
        Choice ← 'N'
}
ENDIF
```

```
IF Choice = 'Y' THEN
{
OUTPUT "Please Purchase Ticket(s)"
OUTPUT "You can purchase ticket(s) only on the day of journey and it must be return ticket"

Flag ← FALSE
REPEAT
{
        IF Flag = TRUE THEN OUTPUT "Wrong number of tickets. Please try again." ENDIF
        OUTPUT "Tickets can be purchased for a single passenger or for a group of 10 to 80 passengers only"
        OUTPUT "How many tickets would you like to buy?"
        INPUT TicketsPurchased
        Flag ← TRUE                                                    //Range Check
}                       //Because tickets can be purchased for a single passenger or a group.
UNTIL ( (TicketsPurchased = 1) OR ( TicketsPurchased > = GroupMin AND TicketsPurchased < = GroupMax) )

CountOfTens ← DIV(TicketsPurchased, 10)        // Calculation of free ticket for every tenth passenger.
                                               // DIV( ): Integer division used to find the quotient after division. Example DIV(23,10) is 2.
Flag ← FALSE
OUTPUT "Select your Departure Train"
REPEAT
        IF Flag = TRUE THEN OUTPUT "Wrong train selected. Please try again." ENDIF
    FOR Count ←1 TO 4                          //Hour(Now): Returns current system hour (integer 0 through 23)
    IF ( Tickets[Count] > = TicketsPurchased AND Time[Count] > Hour(Now) ) THEN
    OUTPUT "Please enter ", Count, " for train at", Time[Count], ":00. This train has ", Tickets[Count], "tickets remaining"
    ENDIF
    NEXT Count
        INPUT DepartTrain                                              //Range Check
        Flag ← TRUE
UNTIL (     (DepartTrain >= 1 AND DepartTrain < = 4) AND
        (Tickets[DepartTrain] > = TicketsPurchased ) AND ( Time[DepartTrain] > Hour(Now) )
        )
```

```
Tickets[DepartTrain] ← Tickets[DepartTrain] – TicketsPurchased        //Calculate and update remaining tickets
FreeTickets[DepartTrain] ← FreeTickets[DepartTrain] + CountOfTens     //Save free tickets for each train

Flag ← FALSE

OUTPUT "Select your Return Train"

REPEAT
        IF Flag = TRUE THEN OUTPUT "Wrong train selected. Please try again." ENDIF
        FOR Count ←5 TO 8                    //Hour(Now): Returns current system hour (integer 0 through 23)
        IF ( Tickets[Count]  > =  TicketsPurchased AND Time[Count] >  Hour(Now) + 1 )THEN
        OUTPUT "Please enter ", Count, " for train at", Time[Count], ":00. This train has ", Tickets[Count], "tickets remaining"
        ENDIF
        NEXT Count
        INPUT ReturnTrain
        Flag ← TRUE                                          //Range Check
UNTIL (     (ReturnTrain > = 5 AND  ReturnTrain < = 8) AND
            (Tickets[ReturnTrain] > =  TicketsPurchased ) AND Time[ReturnTrain] >  (Hour(Now) + 1) )
)

Tickets[ReturnTrain] ← Tickets[ReturnTrain] - TicketsPurchased

FreeTickets[ReturnTrain] ← FreeTickets[ReturnTrain] + CountOfTens


Price ← ( TicketsPurchased – CountOfTens )  *  RoundTripTicketCost     //CONSTANT RoundTripTicketCost ← 50.0


IF CountOfTens > 0 THEN
        OUTPUT "Total Price is $" , TicketsPurchased  *  RoundTripTicketCost
        OUTPUT "Discount is $", CountOfTens  *  RoundTripTicketCost
        OUTPUT "Pay the discounted price of  $", Price, "only and collect your tickets"

ELSE
        OUTPUT "Please pay $", Price, "and collect your ticket"
ENDIF
```

}

ENDIF

Password ← “” //Empty string to initialize Password

OUTPUT “Press E to end the day (only admin) or press any other key to continue”

INPUT Choice

IF Choice = ‘E’ **THEN**

{

REPEAT

OUTPUT “To end the day enter admin password or press N otherwise”

INPUT Password // **“abc123”** is admin password.

UNTIL (Password = “abc123” **OR** Password= “N”**)**

}

ENDIF

UNTIL (Password = “abc123”)

TASK 2 OUTPUT: (Test data: Group of 50, Departure at 9:00 and arrival at 10:00)

```
Departure time is 9:00 and 480 ticket(s) still available in this train
Return time is 10:00 and 480 ticket(s) still available in this train
Departure time is 11:00 and 480 ticket(s) still available in this train
Return time is 12:00 and 480 ticket(s) still available in this train
Departure time is 13:00 and 480 ticket(s) still available in this train
Return time is 14:00 and 480 ticket(s) still available in this train
Departure time is 15:00 and 480 ticket(s) still available in this train
Return time is 16:00 and 640 ticket(s) still available in this train

Dear traveller, do you want to book your journey? (Y/N)Y

Please Purchase Ticket(s)
You can purchase ticket(s) only on the day of journey and it must be return tick
et
Tickets can be purchased for a single passenger or for a group of 10 to 80 passe
ngers only
How many tickets would you like to buy?50
```

```
Select your Departure Train
Please enter 1 for train at 9:00. This train has 480 tickets remaining
Please enter 2 for train at 11:00. This train has 480 tickets remaining
Please enter 3 for train at 13:00. This train has 480 tickets remaining
Please enter 4 for train at 15:00. This train has 480 tickets remaining
1
```

```
Select your Return Train
Please enter 5 for train at 10:00. This train has 480 tickets remaining
Please enter 6 for train at 12:00. This train has 480 tickets remaining
Please enter 7 for train at 14:00. This train has 480 tickets remaining
Please enter 8 for train at 16:00. This train has 640 tickets remaining
5
```

```
Total Price is $ 2500.0
Discount is $ 250.0
Pay the discounted price of  $ 2250.0 only and collect your tickets_
```

```
Press E to end the day (only admin) or press any other key to continue
E

To end the day enter admin password or press N otherwise
_
```

Task 3 – End of the day.
Display the number of passengers that travelled on each train journey and the total money taken for each train journey. Calculate and display the total number of passengers and the total amount of money taken for the day. Find and display the train journey with the most passengers that day.

```
//Task 3 Variable declaration and intialization
INTEGER     Highest ← 0,   TrainID ← 0,   TotalPassengers ← 0,   MultiHigh ← 0

REAL        TotalMoney ← 0.0

FOR Count ←1 TO TotalTrains                          // CONSTANT TotalTrains ←8
        IF Count < 8 THEN                                          (8)
                Passengers ← (480 - Tickets[Count])
        ELSE
                Passengers ← (640 - Tickets[Count])
        ENDIF                                        //CONSTANT OneWayTicketCost ← 25.0

        AmountPerTrain ←( (Passengers – FreeTickets[Count]) * OneWayTicketCost )
                                                                   (25.0)

        OUTPUT "Total passengers in train at " , Time[Count], ":00 ", "were ", Passengers
        OUTPUT "Amount received for train at ", Time[Count], ":00 ", "was $", AmountPerTrain
        Flag ← FALSE
        IF Passengers > Highest THEN
                Highest ← Passengers
                TrainID ← Count
                Flag ← TRUE
                MultiHigh ←1
        ENDIF
        IF  Passengers = Highest  AND Flag = FALSE THEN
                MultiHigh ← MultiHigh +1             //If multiple trains have same number of passengers
        ENDIF
                                                     //Total passengers depart  =  Total passengers return
        IF Count < = 4 THEN TotalPassengers ← TotalPassengers + Passengers ENDIF

        TotalMoney ← TotalMoney +  AmountPerTrain
NEXT Count
```

OUTPUT "Total number of passengers for the day is **",** TotalPassengers

OUTPUT "Total amount of money taken for the day is $**",** TotalMoney

IF MultiHigh = 1 **THEN**

    OUTPUT "Train at **",** Time[TrainID]**, "** has the highest number of passengers today"

    OUTPUT "Total number of passengers in this journey was"**,** Highest

ELSE

    IF TotalPassengers > 0 **THEN**

        OUTPUT MultiHigh**, "** trains have same high number of passengers today"

    ENDIF

ENDIF

TASK 3 OUTPUT: (Test data: Group of 50, Departure at 9:00 and arrival at 10:00)

```
Total passengers in train at 9:00 were 50
Amount received for train at 9:00 was $1125.0
Total passengers in train at 11:00 were 0
Amount received for train at 11:00 was $0.0
Total passengers in train at 13:00 were 0
Amount received for train at 13:00 was $0.0
Total passengers in train at 15:00 were 0
Amount received for train at 15:00 was $0.0
Total passengers in train at 10:00 were 50
Amount received for train at 10:00 was $1125.0
Total passengers in train at 12:00 were 0
Amount received for train at 12:00 was $0.0
Total passengers in train at 14:00 were 0
Amount received for train at 14:00 was $0.0
Total passengers in train at 16:00 were 0
Amount received for train at 16:00 was $0.0
Total number of passengers for the day is 50
Total amount of money taken for the day is $2250.0
2 trains have same high number of passengers today_
```