

Higher Nationals

Internal verification of assessment decisions – BTEC (RQF)

INTERNAL VERIFICATION – ASSESSMENT DECISIONS			
Programme title	Higher National Diploma in Computing		
Assessor	Ms.Gayani Nisansala	Internal Verifier	Mr .Lakindu Premachandra
Unit(s)	Unit 01: Programming		
Assignment title			
Student's name	Ranudi Gayathmie Kariyapperuma		
List which assessment criteria the Assessor has awarded.	Pass	Merit	Distinction
INTERNAL VERIFIER CHECKLIST			
Do the assessment criteria awarded match those shown in the assignment brief?	Y/N		
Is the Pass/Merit/Distinction grade awarded justified by the assessor's comments on the student work?	Y/N		
Has the work been assessed accurately?	Y/N		
Is the feedback to the student: Give details: • Constructive? • Linked to relevant assessment criteria? • Identifying opportunities for improved performance? • Agreeing actions?	Y/N Y/N Y/N Y/N		
Does the assessment decision need amending?	Y/N		
Assessor signature			Date
Internal Verifier signature			Date
Programme Leader signature (if required)			Date
Confirm action completed			

Remedial action taken Give details:			
Assessor signature		Date	
Internal Verifier signature		Date	
Programme Leader signature (if required)		Date	

Higher Nationals - Summative Assignment Feedback Form

Student Name/ID	Ranudi Gayathmie Kariyapperuma - KIR/X-00104243		
Unit Title	Unit 01: Programming		
Assignment Number	01	Assessor	
Submission Date	31.05.2023	Date Received 1st submission	
Re-submission Date		Date Received 2nd submission	
Assessor Feedback:			
<p>LO1. Define basic algorithms to carry out an operation and outline the process of programming an application.</p> <p>Pass, Merit & Distinction Descriptors</p> <p>P1 <input type="checkbox"/> M1 <input type="checkbox"/> D1 <input type="checkbox"/></p>			
<p>LO2. Explain the characteristics of procedural, object-orientated and event-driven programming, conduct an analysis of an Integrated Development Environment (IDE).</p> <p>Pass, Merit & Distinction Descriptors</p> <p>P2 <input type="checkbox"/> M2 <input type="checkbox"/> D2 <input type="checkbox"/></p>			
<p>LO3. Implement basic algorithms in code using an IDE.</p> <p>Pass, Merit & Distinction Descriptors</p> <p>P3 <input type="checkbox"/> M3 <input type="checkbox"/> D3 <input type="checkbox"/></p>			
<p>LO4. Determine the debugging process and explain the importance of a coding standard.</p> <p>Pass, Merit & Distinction Descriptors</p> <p>P4 <input type="checkbox"/> P5 <input type="checkbox"/> M4 <input type="checkbox"/> D4 <input type="checkbox"/></p>			
Grade:	Assessor Signature:		Date:
Resubmission Feedback:			
Grade:	Assessor Signature:		Date:
Internal Verifier's Comments:			
Signature & Date:			

* Please note that grade decisions are provisional. They are only confirmed once internal and external moderation has taken place and grades decisions have been agreed at the assessment board.

Assignment Feedback

Formative Feedback: Assessor to Student			
Action Plan			
Summative feedback			
Feedback: Student to Assessor			
Assessor signature		Date	
Student signature		Date	

Pearson Higher Nationals in Computing

Unit 01: Programming Assignment
01

General Guidelines

1. A Cover page or title page – You should always attach a title page to your assignment. Use previous page as your cover sheet and make sure all the details are accurately filled.
2. Attach this brief as the first section of your assignment.
3. All the assignments should be prepared using a word processing software.
4. All the assignments should be printed on A4 sized papers. Use single side printing.
5. Allow 1" for top, bottom , right margins and 1.25" for the left margin of each page.

Word Processing Rules

1. The font size should be **12** point, and should be in the style of **Time New Roman**.
2. **Use 1.5 line spacing**. Left justify all paragraphs.
3. Ensure that all the headings are consistent in terms of the font size and font style.
4. Use **footer function in the word processor to insert Your Name, Subject, Assignment No, and Page Number on each page**. This is useful if individual sheets become detached for any reason.
5. Use word processing application spell check and grammar check function to help editing your assignment.

Important Points:

1. **It is strictly prohibited to use textboxes to add texts in the assignments, except for the compulsory information. eg: Figures, tables of comparison etc. Adding text boxes in the body except for the before mentioned compulsory information will result in rejection of your work.**
2. Carefully check the hand in date and the instructions given in the assignment. Late submissions will not be accepted.
3. Ensure that you give yourself enough time to complete the assignment by the due date.
4. Excuses of any nature will not be accepted for failure to hand in the work on time.
5. You must take responsibility for managing your own time effectively.
6. If you are unable to hand in your assignment on time and have valid reasons such as illness, you may apply (in writing) for an extension.
7. Failure to achieve at least PASS criteria will result in a REFERRAL grade .
8. Non-submission of work without valid reasons will lead to an automatic RE FERRAL. You will then be asked to complete an alternative assignment.
9. If you use other people's work or ideas in your assignment, reference them properly using HARVARD referencing system to avoid plagiarism. You have to provide both intext citation and a reference list.
10. If you are proven to be guilty of plagiarism or any academic misconduct, your grade could be reduced to A REFERRAL or at worst you could be expelled from the course

Student Declaration

I hereby, declare that I know what plagiarism entails, namely to use another's work and to present it as my own without attributing the sources in the correct way. I further understand what it means to copy another's work.

1. I know that plagiarism is a punishable offence because it constitutes theft.
2. I understand the plagiarism and copying policy of the Edexcel UK.
3. I know what the consequences will be if I plagiarises or copy another's work in any of the assignments for this program.
4. I declare therefore that all work presented by me for every aspects of my program, will be my own, and where I have made use of another's work, I will attribute the source in the correct way.
5. I acknowledge that the attachment of this document signed or not, constitutes a binding agreement between myself and Edexcel UK.
6. I understand that my assignment will not be considered as submitted if this document is not attached to the attached.

ranudigk@gmail.com

Student's Signature:
(Provide E-mail ID)

Date: 31.05.2023
(Provide Submission Date)

Higher National Diploma in Computing

Assignment Brief

Student Name /ID Number	Ranudi Gayathmie Kariyapperuma KIR/X - 00104243
Unit Number and Title	Unit 01: Programming
Academic Year	2021/22
Unit Tutor	Ms.Gayani Nisansala
Assignment Title	Design &Implement a GUI based system using a suitable Integrated Development Environment
Issue Date	09.04.2023
Submission Date	31.05.2023
IV Name & Date	
Submission Format	
<p>This submission will have 3 components</p> <p>1. Written Report</p> <p>This submission is in the form of an individual written report. This should be written in a concise, formal business style using single spacing and font size 12. You are required to make use of headings, paragraphs and subsections as appropriate, and all work must be supported with research and referenced using the Harvard referencing system. Please also provide a bibliography using the Harvard referencing system. (The recommended word count is 1,500–2,000 words for the report excluding annexures)</p> <p>2. Implemented System (Software)</p> <p>The student should submit a GUI based system developed using an IDE. The system should connect with a backend database and should have at least 5 different forms and suitable functionality including insert, edit and delete of main entities and transaction processing.</p> <p>3. Presentation</p> <p>With the submitted system student should do a presentation to demonstrate the system that was developed. Time allocated is 10 to 15 min. Student may use 5 to 10 PowerPoint slides while doing the presentation, but live demonstration of the system is required. Evaluator will also check the ability to modify and debug the system using the IDE.</p>	
Unit Learning Outcomes:	

	LO1. Define basic algorithms to carry out an operation and outline the process of programming an application.
	LO2. Explain the characteristics of procedural, object-orientated and event-driven programming, conduct an analysis of a suitable Integrated Development Environment (IDE). LO3. Implement basic algorithms in code using an IDE. LO4. Determine the debugging process and explain the importance of a coding standard

Assignment Brief and Guidance:	

Activity 1

- A. The Fibonacci numbers are the numbers in the following integer sequence. 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144,

In mathematical terms, the sequence F_n of Fibonacci numbers is defined by the recurrence relation.

$$F_n = F_{n-1} + F_{n-2}$$

- B. Factorial of a non-negative integer, is multiplication of all integers smaller than or equal to n . For example, factorial of 6 is $6*5*4*3*2*1$ which is 720.

$$n! = n * (n - 1) * \dots \dots 1$$

Define what an algorithm is and outline the characteristics of a good algorithm. Write the algorithms to display the Fibonacci series and the factorial value for a given number using Pseudo code. Determine the steps involved in the process of writing and executing a program.

Take a sample number and dry run the above two algorithms. Show the outputs at the end of each iteration and the final output. Examine what Big-O notation is and explain its role in evaluating efficiencies of algorithms. Write the Python program code for the above two algorithms and critically evaluate their efficiencies using Big-O notation.

Activity 2

- 2.1 Explain what is meant by a Programming Paradigm and the main characteristics of Procedural, Object oriented and Event-driven paradigms and the relationships among them. Write small snippets of code as example for the above three programming paradigms using a suitable programming language(s). you also need to critically evaluate the code samples that you have given above in relation to their structure and the unique characteristics.

Activity 3 and Activity 4 are based on the following Scenario.

Ayubo Drive is the transport arm of Ayubo Leisure (Pvt) Ltd, an emerging travel & tour company in Sri Lanka. It owns a fleet of vehicles ranging from cars, SUVs to vans.

The vehicles that it owns are hired or rented with or without a driver. The tariffs are based on the vehicle type. Some of the vehicle types that it operates are, small car, sedan car, SUVs, Jeep (WD), 7-seater van and Commuter van. New vehicle types are to be added in the future.

Vehicle rent and hire options are described below.

1. Rent (With or without driver) – For each type of vehicle rates are given per day, per week and per month. Rate for a driver also given per day. Depending on the rent period the total rent amount needs to be calculated. For example: if a vehicle is rented for 10 days with a driver, total amount to be calculated as follows:

Total rent = weeklyRent x 1 + dailyRent x 3 + dailyDriverCost x 10

2. Hire (with driver only) – These are based on packages such as airport drop, airport pickup, 100km per day package, 200km per day package etc. Standard rates are defined for a package type of a vehicle type if that is applicable for that type of vehicle. For each package maximum km limit and maximum number of hours are also defined. Extra km rate is also defined which is applicable if they run beyond the allocated km limit for the tour. For day tours if they exceed max hour limit, a waiting charge is applicable for extra hours. Driver overnight rate and vehicle night park rate also defined which is applicable for each night when the vehicle is hired for 2 or more days.

Activity 3**Function 1: Rent calculation.**

Return the total rent_value when vehicle_no, rented_date, return_date, with_driver parameters are sent in. with_driver parameter is set to true or false depending whether the vehicle is rented with or without driver.

Function 2: Day tour - hire calculation.

Calculate total hire_value when vehicle_no, package_type, start_time, end_time, start_km_reading, end_km_reading parameters are sent in. Should return base_hire_charge, waiting_charge and extra_km_charge as output parameters.

Function 3: Long tour - hire calculation.

Calculate total hire_value when vehicle_no, package_type, start_date, end_date, start_km_reading, end_km_reading parameters are sent in. Should return base_hire_charge, overnight_stay_charge and extra_km_charge as output parameters.

Write suitable algorithms for vehicle tariff calculation for rents and hires. Ideally 3 functions should be developed for this purpose as above. Use the Visual Studio IDE (using C#.net) to implement the above algorithms and design the suitable database structure for keeping the tariffs for vehicle types and different packages which must be used for implementing the above functions.

Analyze the features of an Integrated Development Environment (IDE) and explain how those features help in application development. Evaluate the use of the Visual Studio IDE for your application development contrasted with not using an IDE.

Activity 4

- 4.1 Design and build a small system to calculate vehicle hire amounts and record them in a database for customer billing and management reporting for Ayubo drive. This includes the completing the database design started in 3.2 and implementing one or more GUIs for vehicle, vehicle type, and package add/edit/delete functions. It essentially requires an interface for hire calculation and recording function described above. Generating customer reports and customer invoices are not required for this course work.
- 4.2 Explain debugging process and the features available in Visual Studio IDE for debugging your code more easily. Evaluate how you used the debugging process to develop more secure, robust application with examples.
- 4.3 Outline the coding standards you have used in your application development. Critically evaluate why a coding standard is necessary for the team as well as for the individual.



Acknowledgement

In the accomplishment of this project successfully, many people have best owned upon me their blessings and the heart pledged support, this time I am utilizing to thank all the people who have been concerned with this project.

I would like to thank Miss Gayani Nisansala whose valuable guidance has been the ones that helped me patch this project and make it full proof success. Their suggestions and their instructions have served as the major contributor towards the completion of the project.

Then I would like to thank my parents and friends who have helped me with their valuable suggestions and guidance has been very helpful in various phases of the completion of the project.

Last but not the least I would like to thank my classmates who

have helped me a lot.

\

Table of Content

Acknowledgement.....	16
Algorithms.....	20
Types of Algorithms	20
• Search Algorithms	20
1) Linear Search Algorithms	21
2) Binary Search Algorithms.....	22
1 st step.....	22
Final step	23
• Sorting Algorithms	24
Types of Sorting Algorithms	24
• Graph Algorithms	25
Characterization of Good Algorithms	26
Fibonacci Series	27
Pseudocode for Fibonacci Series.....	27
Dry run for Fibonacci series.....	28
Python Code for Fibonacci series.....	29
Factorial Numbers	30
Pseudocode for factorial numbers	30
Dry Run for Factorial	31
Python code for Factorial Numbers	31
Steps Involved in the Process of Writing and Executing a Program.....	32
Big O-Notation.....	34
Role in Evaluating Efficiency in Algorithms	34
Analyzing of Fibonacci algorithms using Big O Notation.....	35
Analyzing of Factorial using Big O Notation	36
Programming Paradigm.....	37
• Procedural programming Paradigm.....	37
Characteristics of Procedural Programming Paradigm	37
Advantages	38
Disadvantages.....	39
• Object Oriented Programming Paradigm	40
Characteristics of Object Oriented Programming Paradigm.....	41
Advantages of Object-Oriented Programming Paradigm	42

Disadvantages of Object-Oriented Programming paradigm	44
Event-driven Programming Paradigm.....	45
Characteristics of Event-driven Programming Paradigm.....	46
Advantages of Event-driven Programming Paradigm	47
Disadvantages of Event-driven Programming Paradigm	49
• Sample procedural code in C language	51
• Object Oriented code in Java Programming Language	51
• Event-driven code in Python Language	52
Algorithms for Vehicle Tariff Calculation for Rents and Hires	53
Integrated Development Environment (IDE).....	59
Features of IDE	59
Types of IDE's.....	60
Visual Studio Programming	63
Ayubo Data Added Form in IDE	65
Design and build a small system to calculate vehicle hire amounts	69
Figure 14 : Rent a Vehical	69
Debugging an Application.....	70
Types of Debugging	70
Coding Standards	71

Content of Figures

Figure 1 : Made by author Fibonacci series	27
Figure 2 : Output of Fibonacci series	27
Figure 3 : Made by Author Factorial numbers	29
Figure 4 : Output of Factorial Numbers	30
Figure 5 : sample procedural code in C language	49
Figure 6 : sample object oriented in java programming language	49
Figure 7	53
Figure 8	53
Figure 9	54
Figure 10	54
Figure 11	55
Figure 12	55
Figure 13	56
Figure 14 : Rent a Vehical	67

Activity 01

Algorithms

An algorithm is a detailed, step-by-step collection of instructions or a clearly defined computer process that takes an input or group of inputs and produces the desired output or result. Algorithms are employed in a variety of disciplines, including computer science, mathematics, engineering, and daily life, to solve certain issues, carry out activities, or achieve goals.

Types of Algorithms

- Search Algorithms
- Sorting Algorithms
- Graph Algorithms
- Dynamic Programming
- Greedy Algorithms

These are few types of Algorithms in these algorithms it categorized into another parts. Now author will explain one by one.

• Search Algorithms

Any method that resolves the search problem, namely the need to locate information that has been stored in a data structure or calculated in the search space of a problem domain, using discrete or continuous values, is a search algorithm. There are two types of search algorithms known as ,

Trees, graphs, linked lists, arrays, and other data structures are among the many forms of data structures that searching algorithms can work on. Based on how they operate, how sophisticated they are, and what features of the data they are meant to work with, they can be divided into different categories. There are searching algorithms that use a brute-force method of checking each element one after the other, while other algorithms use more advanced strategies like taking use of the data set's structure or using heuristics to optimize the search.

1) Linear Search Algorithms

To find the desired element, this technique systematically loops through the entire array or list starting at one end. If the element is located, the index is returned; otherwise, -1.'

Ex:- Find 11

2	3	5	8	11	18	23	29
---	---	---	---	----	----	----	----

2 == 11 → No

2	3	5	8	11	18	23	29
---	---	---	---	----	----	----	----

3 == 11 → No

2	3	5	8	11	18	23	29
---	---	---	---	----	----	----	----

5 == 11 → No

2	3	5	8	11	18	23	29
---	---	---	---	----	----	----	----

8 == 11 → No

2	3	5	8	11	18	23	29
---	---	---	---	----	----	----	----

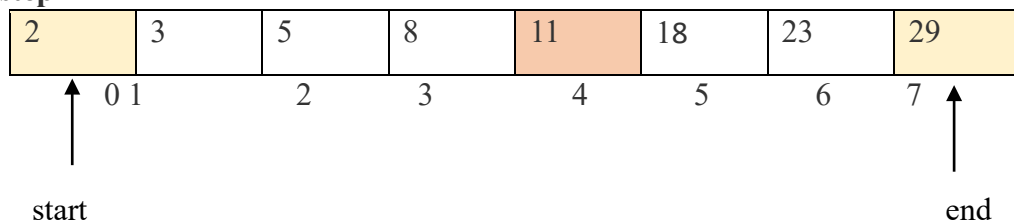
11 → == 11 Yes found

2) Binary Search Algorithms

Among a sorted series of elements, binary search is a traditional algorithm that effectively finds a target value. It operates by halving the search interval on multiple occasions

Ex:- Find 11

1st step

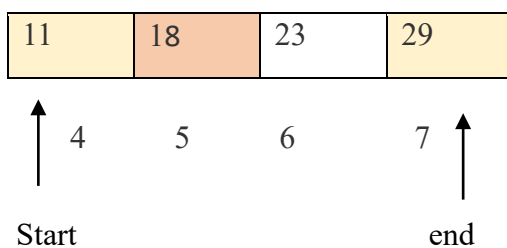


$$\text{Mid} = \frac{\text{start} + \text{end}}{2}$$

$$\begin{aligned}
 &= \frac{0 + 7}{2} \\
 &= \frac{7}{2} \\
 &= 3.5 \\
 &= 3 = 8
 \end{aligned}$$

$8 \leq 11$
 $8 < 11$

2nd step



$$\text{Mid} = \frac{\text{start} + \text{end}}{2}$$

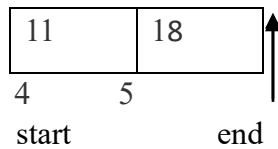
$$\begin{aligned}
 &= \frac{4 + 7}{2} \\
 &= 5.5 \\
 &= 5 = 18
 \end{aligned}$$

$18 \leq 11$
 $18 > 11$

$$= 5.5$$

$$= 5 = 18$$

Final step



$$\text{Mid} = \text{start} + \text{end}$$

$$\begin{aligned} & \frac{2}{=} \frac{4 + 5}{2} \\ & = 4.5 \end{aligned}$$

$$= 4.5$$

$$= 4 = 11$$

$$11 < = > 11$$

$$11 = 11$$

$$\text{return} = 4//$$

• Sorting Algorithms

An element can be arranged in a certain order, usually ascending or descending, using a basic set of techniques called sorting algorithms. These algorithms are essential to computer science and data processing because they allow data sets to be efficiently organized, which facilitates data manipulation, analysis, and search. Sorting algorithms are useful for improving the performance of a wide range of applications since they may work with a variety of data structures, such as lists, arrays, trees, and more.

Different sorting algorithms use various methods and approaches to put the elements back in the order that they should be in. Certain sorting algorithms use comparison-based techniques, which compare components and then swap them around to get the desired arrangement. Divide and conquer tactics are employed by other algorithms, which involve breaking down the data set into smaller subproblems recursively and then combining the sorted subproblems to produce the final sorted sequence.

Types of Sorting Algorithms

1. Bubble Sort

A very basic sorting algorithm is the bubble sort. In order for it to function, it must step through the list several times, compare each pair of adjacent items, and swap them if the order is incorrect. Until the list is sorted, the trip through the list is repeated.

It is not efficient for huge lists, with an $O(n^2)$ time complexity. It does, however, have the benefit of only requiring one pass if the list has previously been sorted.

2. Selection Sort

Choosing Sort places the minimum element at the beginning of the unsorted section after repeatedly locating it. The subarray that has already been sorted and the unsorted subarray are both kept up to date by the method. To shift the smallest element from the unsorted subarray to the end of the sorted subarray, an iteration is performed.

Although it is inefficient for large lists due to its $O(n^2)$ time complexity, it outperforms bubble sort since it requires fewer swaps.

3. Insertion Sort

Placement One item at a time, the final sorted array is constructed using the straightforward sorting algorithm sort. Every element in the list is taken out and placed correctly in the sorted array. As it goes over the list again, bigger elements are moved to the right.

Although it performs well with small data sets or partially sorted lists, its time complexity is $O(n^2)$. For small data sets, it is more efficient than bubble sort and selection sort since it necessitates fewer comparisons and swaps.

• Graph Algorithms

Graph algorithms are a collection of steps or directives for navigating or searching across a graph data structure, and it appears that this is what you are maybe looking at. In many computer science applications, such as network design, routing, and data management, graph algorithms are essential. The following are a few of the most popular graph algorithms:

1. Breadth-First Search

An algorithm for navigating or searching a network is called BFS. Before going on to nodes at the next depth level, it begins at a particular node and investigates every nearby node at the current depth.

BFS is frequently used to visit every node in a graph and determine the shortest path in an unweighted graph.

2. Depth-First Search

Another algorithm for navigating or exploring a network is DFS. It begins at a given node, travels as far as it can along each branch, and then turns around.

DFS is frequently used to explore every node in a connected component and identify cycles in a graph.

Characterization of Good Algorithms

- **Correctness**

For every possible set of valid inputs, the algorithm ought to yield the right result. It ought to effectively resolve the issue that it was intended to handle.

- **Efficiency**

The algorithm's usage of time and memory should be within reasonable bounds. It needs to be tuned to efficiently handle big datasets and intricate calculations.

- **Scalability**

Greater problem sizes should not need a major increase in the amount of time or resources required by the algorithm. Designing it to scale with larger input sizes should be the goal.

- **Adaptability**

The algorithm must be versatile and able to adjust to various scenarios or conditions. It should be able to adapt to changes in the requirements or input data without requiring major adjustments.

- **Simplicity and clarity**

It should be simple to comprehend, apply, and maintain the algorithm. It should be expressed in a clear, succinct style with basic logic that is understandable to other developers.

- **Robustness**

The algorithm should be capable of handling odd or unexpected inputs without breaking down or giving false results. It ought to be equipped with suitable error-handling systems.

- **Optimality**

Under some conditions, the algorithm should yield the best possible outcomes, or as close to the best as possible, given the given limitations. The goal of the algorithm for optimization issues should be to find the best solution possible given the given restrictions.

- Reusability

In order to enable the algorithm to be applied to various issue situations or integrated into bigger systems with minimal modification, it should be developed in a modular and reusable manner.

Fibonacci Series

The Fibonacci sequence is a set of numbers where each number is equal to the sum of the two numbers that came before it. It bears the name Fibonacci, Leonardo of Pisa, who popularized the sequence in the West with the publication of "Liber Abaci" in 1202. Each number after 0 and 1 in the sequence is the sum of the two numbers before it. The recurrence relation defines the Fibonacci number sequence F_n in mathematical terms:

$$F_n = F_{(n-1)} + F_{(n-2)}$$

Usually starting with 0 and 1, the Fibonacci sequence can also start with 1 and 1. Consequently, the order is as follows: 0, 1, 1, 2, 3, 5, 8, 13, 21, and so on.

Pseudocode for Fibonacci Series

Start

Declare variable a, b, c, n, i

Initialize variable a=0, b=1 and i=2

Read n from user

Print a and b Repeat

until $i \leq n$:

c=a+b

print c

a=b, b=c

i=i+1

End

Dry run for Fibonacci series

$f=0, f1=-1, f2=1$ $0 < 7$ $f=f1+f2$ $= -$ $1+1$ $=0$ $f1=f2$ $=1$ $f2=f$ $=0$ $f=0$	$f=0, f1=1, f2=0$ $1 < 7$ $f=f1+f2$ $= 1+0$ $=1 \quad f1=f2$ $=0$ $f2=f$ $=1 \quad f=1$	$f=1, f1=0, f2=1$ $2 < 7$ $f=f1+f2$ $= 0+1$ $=1 \quad f1=f2$ $=1$ $f2=f$ $=1 \quad f=1$
$f=1, f1=0, f2=1$ $3 < 7$ $f=f1+f2$ $= 1+1$ $=2 \quad f1=f2$ $=1$ $f2=f$ $=2 \quad f=2$	$f=2, f1=1, f2=2$ $4 < 7$ $f=f1+f2$ $= 1+2$ $=3 \quad f1=f2$ $=3$ $f2=f$ $=3$ $f=3$	$f=3, f1=2, f2=3$ $5 < 7$ $f=f1+f2$ $= 2+3$ $=5 \quad f1=f2$ $=2$ $f2=f$ $=5$ $f=5$

Python Code for Fibonacci series

Code

```
File Edit Format Run Options Window Help
# Program to display the Fibonacci sequence up to n-th term

nterms = int(input("How many terms? "))
# first two terms
n1, n2 = 0, 1
count = 0
# check if the number of terms is valid
if nterms <= 0:
    print("Please enter a positive integer")
# if there is only one term, return n1
elif nterms == 1:
    print("Fibonacci sequence upto",nterms,":")
    print(n1)
# generate fibonacci sequence
else:
    print("Fibonacci sequence:")
    while count < nterms:
        print(n1)
        nth = n1 + n2
        # update values
        n1 = n2
        n2 = nth
        count += 1
```

Figure 1 : Made by author Fibonacci series

Output

```
File Edit Shell Debug Options Window Help
Python 3.10.6 (tags/v3.10.6:9c7b4bd, Aug 1 2022, 21:53:49) [MSC v.
AMD64] on win32
Type "help", "copyright", "credits" or "license()" for more informa
>>>
= RESTART: C:/Users/HP/OneDrive - NSBM/Esoft HND 10/all assignment
09.2023/programming/fibonanci/fibonanci.py
How many terms? 9
Fibonacci sequence:
0
1
1
2
3
5
8
13
21
>>> |
```

Figure 2 : Output of Fibonacci series

Factorial Numbers

The mathematical function known as the factorial multiplies any positive integer below a non-negative integer. It is represented by the symbol "!" and has the following definition for non-negative integers:

$n!$ is equal to $n * (n-1) * (n-2) * \dots * 3 * 2 * 1$.

For instance:

$0! = 1 * 1 = 1$ $2! = 2 * 1 = 2$ $3! = 3 * 2 * 1$

$= 6$ is equal to $3!$

$4!$ becomes $4 * 3 * 2 * 1 = 24$.

Likewise, $5! = 5 * 4 * 3 * 2 * 1 = 120$.

Applications of factorials can be found in probability, permutations, and combinatorics. For example, the factorial function is used in combinatorics to determine how many ways there are to organize an object. It can also be found in a number of mathematical computations and formulas.

Pseudocode for factorial numbers

Begin

Declare N and F as integer variable.

Initialize F=1.

Enter the value of N.

Check whether $N > 0$, if not then F=1.

If yes then, $F = F * N$

Decrease the value of N by 1 .

Repeat step 4 and 5 until $N = 0$.

Now print the value of F.

The value of F will be the factorial of N(number)

End

Dry Run for Factorial

n=6 fac=fac*n =1*6 =6 n=6-1 =5 fac=6	n=5 fac=fac*n =6*5 =30 n=5-1 =4 fac=30	n=4 fac=fac*n =30*4 =120 n=4-1 =3 fac=120
n=3 fac=fac*n =120*3 =360 n=3-1 =2 fac=360	n=2 fac=fac*n =360*2 =720 n=2-1 =1 fac=720	n=1 fac=fac*n =720*1 =720 n=1-1 =0 fac=720

Python code for Factorial Numbers

Code

```

File Edit Format Run Options Window Help
num = int(input("Enter a number: "))
factorial = 1
if num < 0:
    print("Sorry, factorial does not exist for negative numbers")
elif num == 0:
    print("The factorial of 0 is 1")
else:
    for i in range(1,num + 1):
        factorial = factorial*i
    print("The factorial of",num,"is",factorial)

```

Figure 3 : Made by Author Factorial numbers

Output

```
>>> = RESTART: C:/Users/HP/OneDrive - NSBM/Esoft HND 10/all assignments' final 23 .
09.2023/programming/fibonanci/fibonanci.py
Enter a number: 7
The factorial of 7 is 5040
>>> = RESTART: C:/Users/HP/OneDrive - NSBM/Esoft HND 10/all assignments' final 23 .
09.2023/programming/fibonanci/fibonanci.py
Enter a number: 4
The factorial of 4 is 24
>>> |
```

Figure 4 : Output of Factorial Numbers

Steps Involved in the Process of Writing and Executing a Program

A program's development and operation are ensured by a number of crucial processes that are involved in the writing and execution process.

- **Definition and Analysis of the Problem:**

Recognize the issue that needs to be resolved by the program. Examine the limitations and prerequisites to establish the program's scope.

- **Create Algorithms:**

Use flowcharts or pseudocode to outline the procedures and reasoning required to solve the problem. Select suitable algorithms and data structures to put the answer into practice.

- **Encoding:**

Write the code in the appropriate programming language depending on the designed algorithm. To write and change the code, use an Integrated Development Environment (IDE) or text editor.

- **Gathering and Analyzing:**

If the language is one that can be compiled, then compile the code; if not, then interpret the code. Look for and correct any compilation problems or syntax errors.

- Testing

Run the software through a variety of test cases to make sure everything works as it should. Find and fix any logical mistakes or strange behavior. To guarantee that the program functions properly, find and correct any flaws or problems in the code.

- Debugging

Utilize the IDE's debugging tools to find and fix issues. Include comments in the code to describe its use, functionality, and any intricate logic.

- Documentation

If required, write program user manuals or technical documentation.

Big O-Notation

In computer science, big O notation is a mathematical notation used to express an algorithm's complexity or performance. As the amount of input data grows, it offers an upper constraint on the algorithm's asymptotic behavior, especially with regard to time and space complexity.

It is written as $O(f(n))$, where $f(n)$ is a function that expresses how quickly the algorithm grows in relation to the size of the input, indicated by the letter n . It aids in comprehending how an algorithm's time or space needs rise as input size does.

Role in Evaluating Efficiency in Algorithms

Big O notation is essential for assessing an algorithm's efficiency. It functions as a common, standardized metric for evaluating and contrasting the effectiveness of various algorithms, especially when it comes to their temporal and spatial complexity. Big O notation plays several important roles in assessing the efficacy of algorithms, including:

- Performance Comparison

The scalability and efficiency of various algorithms can be compared thanks to Big O notation. It offers a standard vocabulary to describe how an algorithm's runtime or memory consumption increases as the size of the input increases.

- Algorithm Selection

Based on an algorithm's performance characteristics, developers can choose the best algorithm for a given problem by comprehending the Big O notation of several algorithms. They can pick algorithms that show greater efficiency for the particular task and size of input.

- Finding Bottlenecks

Big O notation aids in locating possible bottlenecks or problems with an algorithm's performance. It enables programmers to identify potential inefficiencies in the algorithm or places where performance might be enhanced by applying improvements.

- Optimization Strategies

It offers information on how to use optimizations to improve an algorithm's efficiency. Big O notation provides information that developers can utilize to apply optimization strategies and enhance the algorithm's overall performance.

Analyzing of Fibonacci algorithms using Big O Notation

When n is the input value, the temporal complexity of the Fibonacci method can be written as $O(2^n)$. The time needed to compute the Fibonacci sequence increases exponentially with the input value. For higher values of n , the recursive Fibonacci algorithm is extremely inefficient due to its exponential time complexity.

By avoiding unnecessary calculations and storing the answers of subproblems, dynamic programming or memoization techniques can be utilized to optimize the temporal complexity. This method is substantially more efficient for bigger values of n since it can reduce the time complexity to linear time $O(n)$.

Practically speaking, depending on the computing environment, the naïve recursive Fibonacci algorithm cannot be used to calculate big Fibonacci numbers due to its exponential time complexity, especially for values of n greater than 30 or 40. For actual use cases, it is therefore imperative to construct optimized variants of the Fibonacci method.

Analyzing of Factorial using Big O Notation

The function iterates from 1 to n in this iterative implementation, multiplying each number as it goes. This factorial algorithm has an $O(n)$ time complexity, where n is the input value. The time needed to compute the factorial grows linearly with the input value.

Because the intermediate result and the loop variable only need a fixed amount of extra space to be stored, the space complexity of this technique is also $O(1)$.

If the input values are not very large, the factorial algorithm is efficient for the majority of real-world use situations due to its linear time complexity. However, the factorial can rise quickly for very large values of n , which could cause computational problems and an overflow of integer data types. Under such circumstances, the hardware limits or the type of data used to store the results may have an effect on the algorithm's performance.

One can use big integer libraries or other methods to manage arbitrarily large numbers to handle huge values of n more effectively. Depending on the implementation, this may result in increased time and space difficulties, but it can guarantee accurate factorial computation for big inputs.

Activity 2

Programming Paradigm

Programming paradigms are basic programming styles or approaches that are defined by a set of ideas, procedures, and principles that direct the composition and arrangement of computer programs. Programming paradigms are distinct ways of approaching and resolving issues; each paradigm emphasizes specific methods and approaches to software design and implementation. The object-oriented, functional, procedural, and event-driven programming paradigms are a few popular programming styles.

Characteristics of Programming Paradigm

- **Procedural programming Paradigm**

A programming paradigm known as procedural programming concentrates on writing routines or procedures that carry out particular tasks. It highlights the sequential set of instructions that the computer needs to follow in order to complete a task. Programs written in procedural programming are organized around functions, subroutines, or methods—procedures that manipulate data.

Characteristics of Procedural Programming Paradigm

- Procedure Calls

To carry out the intended tasks, the software is broken up into small, self-contained processes that can be called in a particular order.

- Emphasis on Procedures

It emphasizes the use of procedures to do tasks, improving the modularity and reusability of code.

Procedural programs are carried out in a sequential fashion, with a distinct transfer of control from one instruction to the next.

- Data Structures and Variables

It uses data structures and variables, including arrays, lists, and records, to store and manage data.

- Structured Programming

To improve the readability and maintainability of code, it promotes the use of control structures like loops and conditionals, among other structured programming techniques.

Advantages and Disadvantages of Programming Paradigm

Advantages

- **Simplicity:** It's not too difficult to comprehend and use procedural programming. It takes a simple top-down approach, which makes it easier for programmers to understand how the program flows.
- **Reusability:** Code modularity and reusability are encouraged by the program's ability to reuse procedures and functions. By creating specialized methods that may be used in other areas of the software, developers can cut down on redundancy and enhance maintainability.
- **Efficiency:** Memory utilization and execution speed are two areas where procedural programming languages excel. They are appropriate for creating applications that are crucial to performance because they give the developer direct control over the hardware resources.
- **Hardware Control:** Procedural programming is appropriate for jobs requiring direct hardware interaction because it provides low-level control over hardware resources.

- **Unambiguous Task Division:** Procedural programming makes it easier to divide tasks into more manageable, smaller procedures. Because problems can be localized inside particular procedures, this technique streamlines the debugging and maintenance procedures.
- **Wide Support:** Procedural programming is compatible with a wide range of programming languages, which makes it useful and accessible in a variety of settings and development environments.
- **Optimization:** Using procedural programming, programmers can enhance the effectiveness and performance of their code. An organized procedural structure makes it simpler to find and remove inefficiencies in the software.

Disadvantages

- **Problems Managing State**

Using global variables is a common practice in procedural programming, and this can cause problems with managing the state of the program. Global variables can provide unexpected side effects and make it difficult to monitor and manage how data moves through the application.

- **Lack of Scalability**

Procedural programming may not scale well as systems get larger and more complicated, which can result in hard to comprehend and maintain code. In large-scale applications, managing dependencies and interactions between different operations can become more difficult.

- **Limited Abstraction**

Complex real-world things and relationships may be more difficult to describe with procedural programming since it may not offer the same amount of abstraction as other paradigms. This restriction may lead to less understandable and straightforward code, particularly for intricate systems.

- **Poor Modularity Support:**

Procedural programming may not offer the same degree of encapsulation and modularity as other paradigms, although supporting a certain amount of modularity through functions and procedures. This restriction may cause problems with the organization, upkeep, and reuse of the code.

Reusing code in procedural programming can be difficult, particularly when working with closely related routines. In many situations, modifying one section of the software can need modifying several other sections as well, increasing the likelihood of errors and problems creeping in.

- **Complexity as Programs Expand**

Procedural programming can result in intricate, jumbled code structures as programs get bigger. Large-scale program management in a procedural paradigm might make it harder to maintain, debug, and grow the codebase.

- **Limited Code Reusability**

Procedural programming has a limit on how much code can be reused, and it may not be as modular and reusable as other programming paradigms like object-oriented programming. This restriction may lead to duplication code and complicate codebase management and updates.

- **Object Oriented Programming Paradigm**

The paradigm for programming known as object-oriented programming (OOP) is based on the idea of "objects," which are instances of classes that have both data and methods. It places emphasis on structuring code into interactive objects that make it possible to model real-world entities and their connections. The concepts of polymorphism, inheritance, encapsulation, and data abstraction are all supported by OOP.

Characteristics of Object Oriented Programming Paradigm

Classes and Objects

The foundation of object-oriented programming is the idea of classes. Objects are instances of classes that contain data and behavior, and classes are blueprints that specify the structure and behavior of objects.

Abstraction

key component of object-oriented programming (OOP). It allows developers to represent real-world entities as objects with certain properties and behaviors, all without disclosing the implementation details underneath.

Encapsulation

Object-oriented programming (OOP) encourages encapsulation, which is the grouping of data and methods that work with the data into a single unit, limiting external access to an object's internal state.

Inheritance

Classes can inherit characteristics and actions from other classes because to OOP's support for inheritance. This feature makes it possible to establish hierarchical links between classes, which encourages the reuse of code and the development of a class hierarchy.

Polymorphism

In object-oriented programming, polymorphism is the capacity of an object to assume multiple forms or to behave in various ways depending on the situation. By using a common interface, it enables various classes to be viewed as instances of the same class.

Modularity

By dividing large systems into smaller, easier-to-manage modules (classes), object-oriented programming (OOP) promotes modularity. This encourages the creation of scalable applications, maintainability, and code reuse of externally-sourced objects.

Advantages of Object-Oriented Programming Paradigm

Code Reusability

OOP encourages code reusability by allowing developers to construct new classes based on pre-existing ones through the concept of inheritance. With this functionality, development time and effort are greatly decreased, resulting in codebases that are more effective and manageable.

Modularity

By using classes to create modular programs, OOP makes it easier for developers to divide large, complex systems into smaller, easier-to-manage modules. Code organization, maintenance, and reusability are all improved by this modularity.

Abstraction

By removing superfluous implementation details, OOP allows the representation of real-world elements in software systems through the concepts of objects and classes. This improves the code's readability and maintainability while streamlining the development process.

Encapsulation

OOP encourages data encapsulation, which makes it possible to combine procedures and data into a single unit (class). This feature ensures data integrity and security within the application by shielding the data from outside intervention and misuse.

Efficiency and Speed

Because OOP allows developers to use pre-existing classes and libraries, they can write less new code from scratch, which can contribute to increased efficiency and speed in the development process. Faster development cycles and more effective resource use are the outcomes of this.

Simple Maintenance

When the encapsulation concept is adhered to, modifications made to one section of the codebase do not impact other sections, making OOP programs simpler to maintain and update. By doing this, the possibility of adding problems and mistakes during maintenance is decreased.

Flexibility and Scalability

Object-Oriented Programming (OOP) offers developers the opportunity to add and modify features to software systems without changing the entire codebase. This flexibility makes software systems easier to manage and extend. This scalability is necessary to meet changing user needs and business requirements.

Simplicity and Understandability

OOP encourages the representation of real-world objects and their interactions in a straightforward and intuitive manner, which facilitates developer comprehension and interaction with the code. This clarity improves the software system's reading and comprehension.

Disadvantages of Object-Oriented Programming paradigm

Complexity

Because of its many features and ideas, including inheritance, polymorphism, and encapsulation, object-oriented programming (OOP) can be hard, especially for novices. Because of this complexity, it may be difficult for inexperienced programmers to understand and successfully use OOP ideas.

Overhead

Because OOP involves more levels of abstraction and indirection in the method invocation process, it may result in performance overhead. This expense, particularly in contexts with limited resources, might affect the application's execution speed and memory utilization.

Steep Learning Curve

OOP has a steep learning curve that makes it necessary for developers to fully comprehend different concepts before using them in a productive way. It could take some time and effort to fully understand OOP best practices and principles, particularly for programmers switching from other programming paradigms.

Big File Sizes

Compared to procedural programming, object-oriented programming (OOP) frequently results in bigger file sizes and memory footprints. The main cause of this is the addition of extra information and object-oriented code constructs, which can make the application larger overall.

Stiff Structure

OOP can occasionally force a stiff structure on the software system's design, making it difficult to expand or modify the codebase. Changes to one section of the code occasionally

need changes to several other sections as well, increasing the effort required to create the code and maybe increasing the risk of inconsistencies.

Inefficient for Some Applications

While object-oriented programming (OOP) is a good way to create large software systems, it might not be the best choice for some applications that need low-level and high-performance hardware.

Difficulty in Designing Parallel systems

Because handling shared state and synchronization across several objects can be difficult and error-prone, object-oriented programming (OOP) can make it difficult to create and implement concurrent and parallel systems. Data consistency and racial conditions may suffer as a result.

Memory Management Overhead

OOP languages frequently call for memory management strategies such as garbage collection, which can result in memory management overhead and possibly affect the application's overall performance.

Event-driven Programming Paradigm

A programming paradigm known as "event-driven programming" is predicated on the idea of events and event handlers. According to this paradigm, human input (such as keystrokes or mouse clicks), sensor outputs, or messages from other threads or programs control how the program flows

Characteristics of Event-driven Programming Paradigm

Event Handlers

Specialized routines or functions created to handle certain events are called event handlers, and event-driven programming depends on them. Certain events, such as user activities, system notifications, or information from outside sources, cause certain event handlers to be triggered.

Asynchronous Processing

Asynchronous processing is frequently used in event-driven systems, enabling the software to carry out tasks while it waits for replies or events to happen. This improves the user experience overall by allowing the system to continue responding to different inputs and events.

Event Queues

Event-driven systems use queues of incoming events to organize and rank them. In order to ensure that events are handled promptly and efficiently, they are usually queued and processed in the order that they are received.

Event Loop

An event loop is frequently used in event-driven programming. It keeps an eye on the event queue all the time to see when new events arrive. To keep the software responsive and able to handle several events at once, the event loop sends out the proper event handlers to handle each event.

Callback Functions

To handle asynchronous actions, callback functions are widely employed in event-driven programming. These functions are called once particular tasks or events are finished and

are supplied as arguments to other functions. In reaction to particular events, the software can carry out extra operations or activities thanks to callback functions.

Emphasis on User Interaction

Event-driven programming is ideally suited for programs that make use of graphical user interfaces (GUIs), as it frequently highlights user interactions. Mouse clicks, keyboard inputs, and menu selections are examples of user activities.

Advantages of Event-driven Programming Paradigm

Responsiveness

Event-driven programming ensures that programs are sensitive to both external events and user inputs, which makes for a seamless and engaging user experience. Because event-driven systems are asynchronous, many processes can run simultaneously without stalling the main thread, keeping the application from becoming unresponsive.

Scalability

Event-driven systems have a great degree of scalability and can effectively manage many users and events at once. Because of its scalability, event-driven programming is a good fit for systems that need to analyze data in real-time and can adapt to changing user demands and workloads.

Modularity

By enabling developers to design and incorporate unique event handlers for certain events or functionality, event-driven programming fosters modularity. It is simpler to update and expand the functionality of the program because to this modularity, which improves code structure and maintainability without changing the entire codebase.

User Interface Development

Creating interactive applications and graphical user interfaces (GUIs) is a great use case for event-driven programming. With the help of the event-driven technique, developers can create user interfaces that react instantly to keystrokes, mouse clicks, and menu selections, giving users a smooth and interesting experience.

Concurrent Processing

Event-driven systems are capable of managing several tasks and events at once, which makes effective use of the system's resources possible. Due to the application's ability to handle numerous tasks at once, throughput inside the system is increased and overall performance is improved.

Development Process Simplified

By allowing developers to concentrate on managing particular occurrences and creating event-driven workflows, event-driven programming streamlines the development process. This method simplifies the creation of intricate systems and applications and lessens the difficulty of handling synchronous

Disadvantages of Event-driven Programming Paradigm

Complexity of Control Flow

When using event-driven programming, particularly in applications that have a large number of event handlers and callbacks, complicated control flow and program logic may result. Controlling the flow of events and organizing event-driven processes can get complicated, leading to hard to maintain and troubleshoot code.

Difficulties with Asynchronous Programming

Asynchronous programming is a basic feature of event-driven programming, but it can present challenges like race situations, deadlocks, and problems with shared resources. It might be difficult and time-consuming to handle asynchronous processes and maintain data consistency.

Debugging and Error Handling

When working with several concurrent events and asynchronous activities, debugging event-driven programs can be extremely difficult. Finding the underlying cause of mistakes and following the series of actions that result in certain problems

Event Cascading and Dependencies

In event-driven systems, when the occurrence of one event sets off a series of subsequent events, event cascading and dependencies may transpire. Complex event dependency management and unintentional event cascade prevention can result in unpredictable behavior and possible performance problems.

sophisticated Event Handling Logic

It can be difficult to create and maintain sophisticated event handling logic, particularly for systems that call for interconnected event-driven workflows and complex event processing.

It takes considerable thought and preparation to create reliable, scalable event-driven systems that can effectively handle intricate event interactions.

Performance Overhead

Because event handlers are executed and events are continuously monitored, event-driven systems may result in performance overhead. The performance can be affected by managing event queues and handling numerous events at once.

Snippets of code

- Sample procedural code in C language

```
#include <stdio.h>
int main() {

    int number1, number2, sum;

    printf("Enter two integers: ");
    scanf("%d %d", &number1, &number2);

    // calculating sum
    sum = number1 + number2;

    printf("%d + %d = %d", number1, number2, sum);
    return 0;
}
```

Figure 5 : sample procedural code in C language

Some features of the procedural programming paradigm, like local variables and predefined functions, are visible in the code snippet above. It is evident that the variables `sum`, `number 1`, and `number 2` are local. The functions `"printf()"` and `"scanf()"` are predefined.

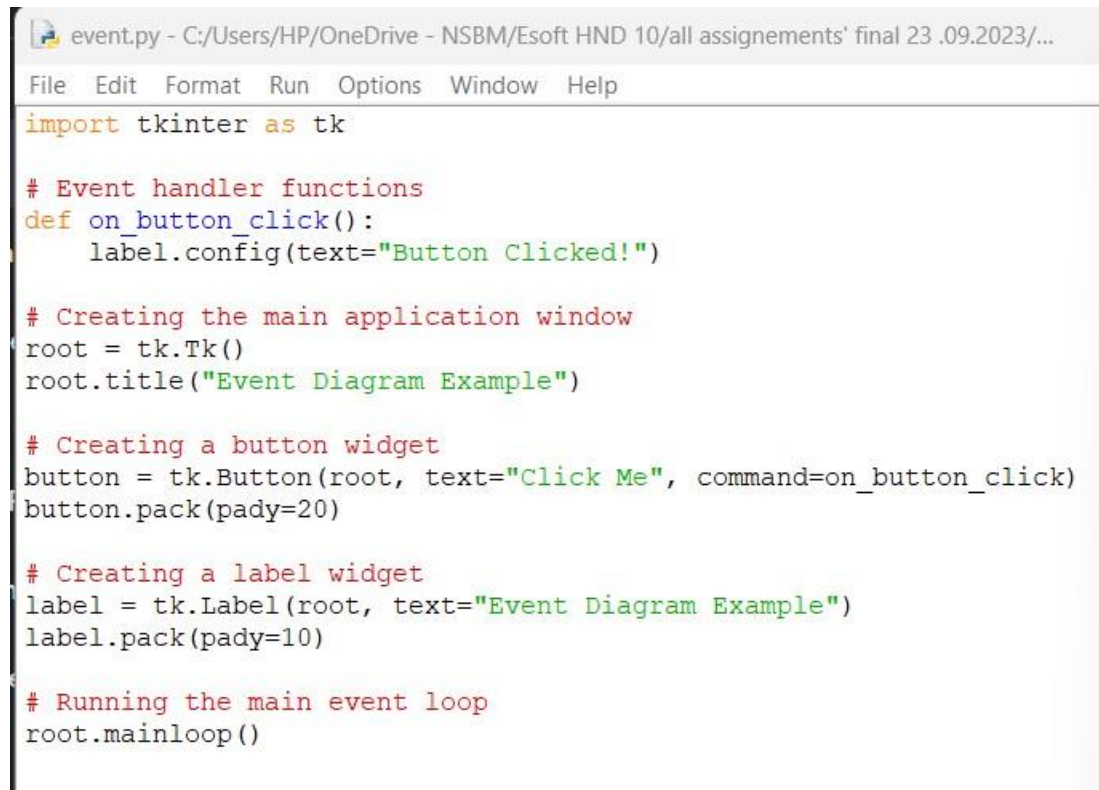
- Object Oriented code in Java Programming Language

```
class Employee{
    float salary=40000;
}
class Programmer extends Employee{
    int bonus=10000;
    public static void main(String args[]){
        Programmer p=new Programmer();
        System.out.println("Programmer salary is:"+p.salary);
        System.out.println("Bonus of Programmer is:"+p.bonus);
    }
}
```

Figure 6 : sample object oriented in java programming language

The Java programming language is used to demonstrate the inheritance feature of the object-oriented programming paradigm in the code snippets above.

- **Event-driven code in Python Language**



```
event.py - C:/Users/HP/OneDrive - NSBM/Esoft HND 10/all assignments' final 23 .09.2023/...
File Edit Format Run Options Window Help
import tkinter as tk

# Event handler functions
def on_button_click():
    label.config(text="Button Clicked!")

# Creating the main application window
root = tk.Tk()
root.title("Event Diagram Example")

# Creating a button widget
button = tk.Button(root, text="Click Me", command=on_button_click)
button.pack(pady=20)

# Creating a label widget
label = tk.Label(root, text="Event Diagram Example")
label.pack(pady=10)

# Running the main event loop
root.mainloop()
```

With Python and the Tkinter package, the user creates a basic GUI application. There is a button and a label on the application. The `on_button_click` method is called when the button is clicked, changing the label's text to "Button Clicked!"

Activity 3

Algorithms for Vehicle Tariff Calculation for Rents and Hires

```
//Rent calculation
1 reference
void Payments()
{
    try
    {
        DateTime from = rfromDatePicker.Value;
        DateTime to = rtoDatePicker.Value;
        int days = 0;
        int noOfMonths = 0;
        int noOfWeeks = 0;
        int noOfDays = 0;
        float driverCost = 0;
        float dailyRent = 0;
        float weeklyRent = 0;
        float monthlyRent = 0;
        float payment = 0;

        days = (to - from).Days + 1;
        noOfMonths = days / 30;
        noOfWeeks = (days - (noOfMonths * 30)) / 7;
        noOfDays = days - ((noOfMonths * 30) + (noOfWeeks * 7));
        driverCost = float.Parse(rdriverchargeTxt.Text);
        dailyRent = float.Parse(rdailyTxt.Text);
        weeklyRent = float.Parse(rweeklyTxt.Text);
        monthlyRent = float.Parse(rmonthlyTxt.Text);
        payment = (dailyRent * noOfDays) + (weeklyRent * noOfWeeks) + (monthlyRent * noOfMonths) + (days * driverCost);

        rpaymentLbl.Text = "Rs. " + payment.ToString();
    }
    catch (Exception ex)
    {
        MessageBox.Show("Error" + ex);
    }
    finally
    {
        conn.Close();
    }
}
```

```
1 reference
void HirePayment()
{
    //Airport Drop
    //Airport Pickup
    //100KM
    //200KM

    string category = hcategory.Text;
    //int perKm = int.Parse(category);
    switch (category)
    {
        case "Airport Drop":
            AirDropandpickup();
            break;
        case "Airport Pickup":
            AirDropandpickup();
            break;
        case "100KM":
            KM100and200();
            break;
        case "200KM":
            KM100and200();
            break;
        default:
            MessageBox.Show("Please select Hire type");
            break;
    }
}
```

```
//Airport drop and pickup calculation
2 references
void AirDropandpickup()
{
    float startMeter = float.Parse(hStart.Text);
    float endMeter = float.Parse(hEnd.Text);
    float km = endMeter - startMeter;

    int perKm = int.Parse(hPerKm.Text);
    float waitingCha = float.Parse(hWaitingCharge.Text);
    waitingCha = waitingCha * 75;

    float payment = (km * perKm) + waitingCha;

    hPayment.Text = "Rs " + payment;
}
```


Use the visual studio IDE (Using c#.net) to implement the above algorithm.

- Login Form

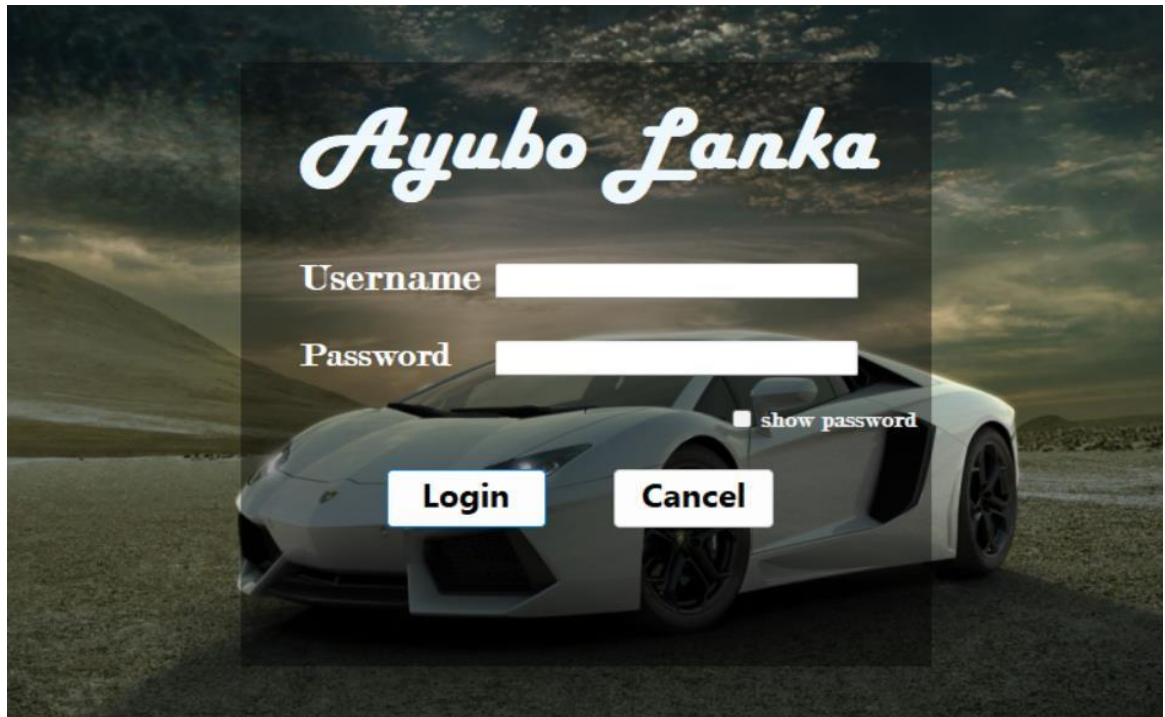


Figure 7

- Main Menu



Figure 8

- Rent a Vehicle Form



Ayubo Lanka

Rent a Vehical

Vehical ID
 Driver ID
 Customer ID


Date From
 Date To

Total Rent

Calculate Bill
 Save Update
 Cancel

Figure 9

- Package Form



Ayubo Lanka

Packages

Pakage ID Save
 Type Update
 Price
 Max(km)
 Max(hours) Cancel

Figure 10



Atyubo Lanka

Packages

Package ID

Type

Price

Max(km)

Max(hours)

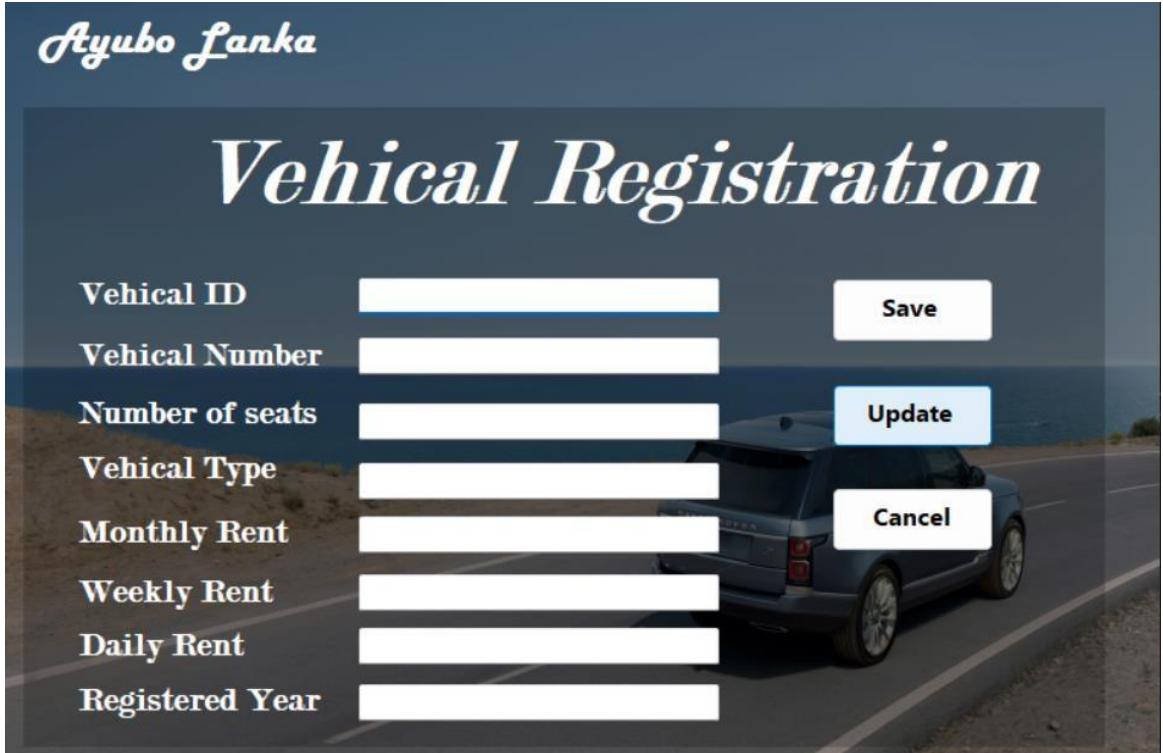
Save

Update

Cancel

Figure 11

- Vehicle Registration Form



Atyubo Lanka

Vehical Registration

Vehical ID

Vehical Number

Number of seats

Vehical Type

Monthly Rent

Weekly Rent

Daily Rent

Registered Year

Save

Update

Cancel

Figure 12

- Driver Details Form

Ayubo Lanka

Driver Details

Name

NIC No

Date of Birth

Phone Number

Address

Driver ID

License No

Save

Update

Cancel

Figure 13

Integrated Development Environment (IDE)

A software program called an Integrated Development Environment (IDE) offers a wide range of features and capabilities to make the entire software development process easier. It functions as a centralized platform for software authoring, modification, compilation, debugging, and release. By offering a consistent, user-friendly interface that combines a variety of programming tools and functionalities, integrated development environments (IDEs) are intended to boost developers' productivity.

Features of IDE

- **Code Editor:** An IDE comes with a feature-rich code editor that can format code, highlight syntax, and do auto-completion. It makes the coding process more convenient and error-free by enabling developers to write and edit code efficiently.
- **Compilers and Interpreters:** To enable debugging and real-time code execution, integrated development environments (IDEs) frequently incorporate compilers and interpreters. With the help of this integration, developers can write, execute, and debug code all within the IDE environment, receiving instant feedback on any mistakes or problems.
- **Debugger:** IDEs have built-in debuggers that let developers find and correct coding issues as they're being developed. Breakpoints, step-by-step execution, variable inspection, and stack tracing are examples of debugging capabilities that make it easier for developers to locate and fix errors quickly.
- **Integration of IDEs with Version Control Systems:** Code versioning, branching, merging, and collaboration are made easier by IDEs' frequent integration with version control systems like Git, SVN, or Mercurial. Through this interface, developers can efficiently monitor and manage codebase changes, guaranteeing a productive development process.
- **Graphical User Interface (GUI) Builder:** Developers can design and construct graphical user interfaces for programs graphically with the use of layout tools or GUI builders, which are available in many IDEs. With the use of these technologies, developers can

create complicated user interfaces more quickly and see a real-time preview of the design.

- **Project Management Tools:** IDEs include project management tools that assist developers in setting up and overseeing resource management, task tracking, file organization, and other project-related activities. These technologies improve team member coordination and collaboration on projects.

Types of IDE's

IDEs for general purposes:

Microsoft's all-inclusive IDE, Visual Studio, supports a number of programming languages, including C#, C++, and .NET.

Eclipse: An open-source integrated development environment, mostly for Java programming, but it can also support other languages with plugins.

NetBeans: An open-source, free integrated development environment (IDE) that supports PHP, JavaScript, and Java.

A well-liked integrated development environment (IDE) for Java development that also supports Kotlin, Groovy, and Scala is IntelliJ IDEA.

Specific language IDEs:

Xcode: An integrated development environment (IDE) designed especially for creating macOS and iOS apps in Swift and Objective-C.

Based on IntelliJ IDEA and specifically designed for Android programming, Android Studio is an official integrated development environment.

PyCharm: An integrated development environment (IDE) for Python that provides support for data analysis, scientific computing, and web development.

RStudio is an IDE created

IDEs for web development:

WebStorm: An IDE with support for multiple web development frameworks, created especially for JavaScript, HTML, and CSS.

Sublime Text: An IDE-like text editor with a wealth of functionality and substantial web development plugin support.

Atom: An IDE-like text editor that can be customized and used for web development, thanks to a number of plugins.

Visual Studio Code: An extensible code editor that is both lightweight and strong, supporting a wide range of programming languages, allows for significant customisation.

Explain how those features help in application development.

Visual Studio (VS) was first launched in 1997 under the moniker Visual Studio 97, which had a version number of 5.0. Released on March 7, 2017, Visual Studio 15.0 is the most recent version. Another name for it is Visual Studio 2017. The gave support.Framework for Net Visual Studio version ranges from 3.5 to 4.7 as of late. While Java was supported in earlier iterations of Visual Studio, it is not supported in the most recent version.

An integrated development environment is called Visual Studio (IDE). It is employed in the development of mobile apps, websites, web services, and computer programs. Microsoft software development platforms, including Windows API, Windows Form, Windows Presentation Foundation, Windows Store, and others, are used by Visual Studio. Both native and managed code can be produced by it.

As long as a language-specific service is available, Visual Studio can support 36 distinct programming languages and enable the code editor and debugger to support almost any programming language to varied degrees. C, C++, C++/CVI, Visual Basic.net, C#F#, Javascript, Typescript, XML, XSLT, HTML, and CSS are among the built-in languages. Plug-ins provide support for additional languages, including M, Python, Ruby, Node Js, and others. In the past, support was provided for Java and J#.The Community edition of Visual Studio, which is the most basic version, is offered without charge. The tagline "Free, fully-featured IDE for students, open-source, and individual developers" refers to the Visual Studio Community edition.

You can edit, debug, and build code using the Visual Studio integrated development environment, and then publish the results.

Compilers, code completion tools, graphical designers, and a ton of other capabilities are all included in Studio to make the software development process easier.

1. You may examine, navigate, and manage your code files using Solution Explorer (upper right). Solution Explorer groups files into solutions and projects to assist with code organization.

2. File contents are shown in the Editor Window (middle), where you'll probably spend most of your time. This is where you can construct a user interface, such a window with text boxes and buttons, or update code.

3. With version control systems like Git and Team Foundation version control, you may exchange code with others and keep track of work items via Team Explorer (bottom right).

Visual Studio Programming

Visual programming is a programming paradigm that allows programmers to create software using graphical elements, such as visual expressions, flowcharts, and diagrams, rather than using textual code. This approach is often used to simplify programming for non-expert users and to create a more intuitive development process. Visual programming languages (VPLs) provide a way for users to create programs by manipulating graphical elements rather than by specifying them textually.

There are several types of visual programming:

Flow-based programming (FBP): FBP is a specific form of dataflow programming based on the idea of defining a program as a network of "black box" processes, which exchange data through predefined connections. It allows developers to visually represent the flow of data through a system.

Block-based programming: This approach involves dragging and dropping blocks or visual elements to represent programming concepts. Examples include Scratch, Blockly, and MIT App Inventor, which are widely used in educational settings to introduce programming concepts to beginners.

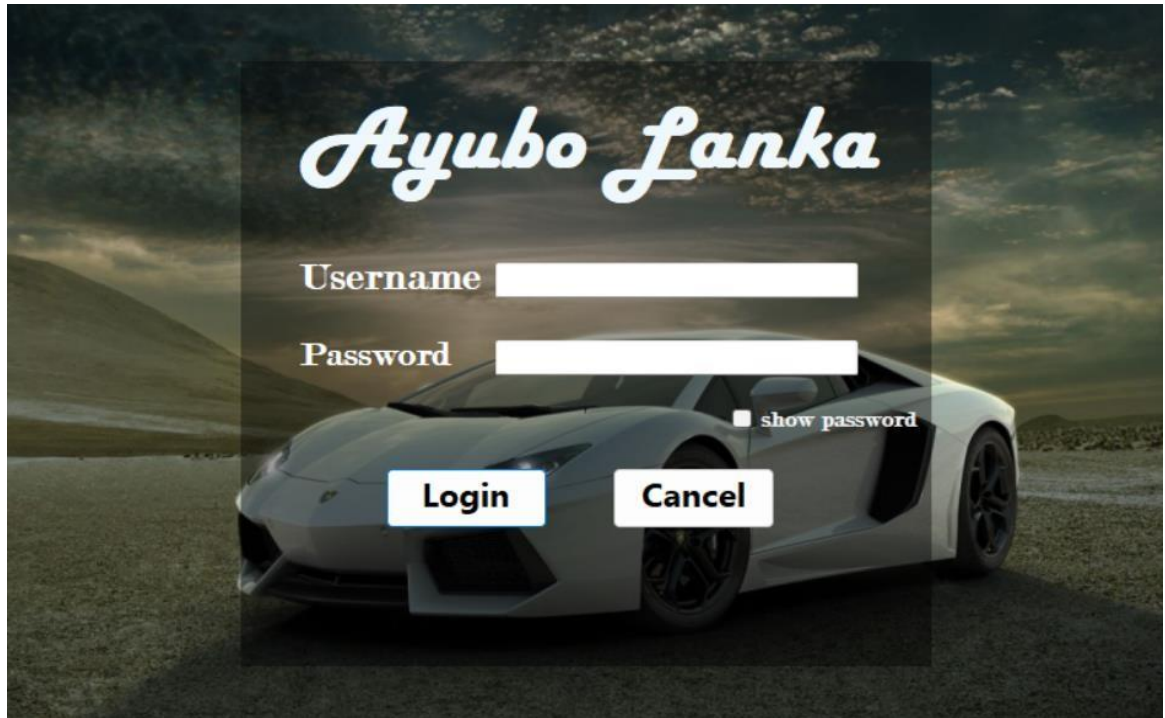
Dataflow programming: This type of visual programming focuses on the flow of data through a system. Programs are represented as a network of interconnected nodes, with each node performing a specific operation on the data.

Visual scripting: This is often used in game development and 3D modeling software, where users can create complex behaviors and interactions by connecting predefined visual

elements. Unity's visual scripting tool, Bolt, and Unreal Engine's Blueprint system are examples of popular visual scripting tools used in game development

Activity 4**Ayubo Data Added Form in IDE**

- Login Form



- Main Menu



- Rent a Vehicle Form



Ayubo Lanka

Rent a Vehical

Vehical ID
 Driver ID
 Customer ID

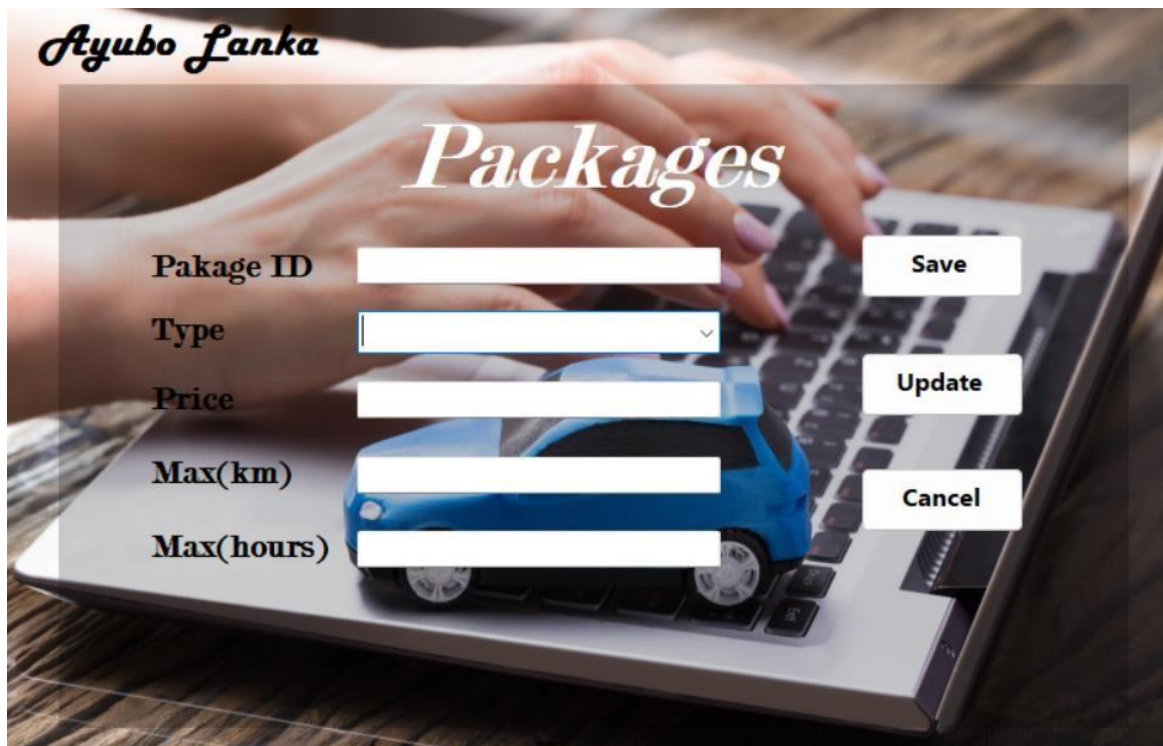
Date From

Date To

Total Rent

Calculate Bill
 Save Update
 Cancel

- Package Form



Ayubo Lanka

Packages

Pakage ID
 Type
 Price
 Max(km)
 Max(hours)

Save
 Update
 Cancel

Ayubo Lanka

Packages

Package ID	<input type="text"/>	Save
Type	<input type="text"/>	Update
Price	Day tour Long tour	
Max(km)	<input type="text"/>	Cancel
Max(hours)	<input type="text"/>	

- Vehicle Registration Form

Ayubo Lanka

Vehical Registration

Vehical ID	<input type="text"/>	Save
Vehical Number	<input type="text"/>	Update
Number of seats	<input type="text"/>	
Vehical Type	<input type="text"/>	Cancel
Monthly Rent	<input type="text"/>	
Weekly Rent	<input type="text"/>	
Daily Rent	<input type="text"/>	
Registered Year	<input type="text"/>	

- Driver Details Form



Ayubo Lanka

Driver Details

Name

NIC No

Date of Birth

Phone Number

Address

Driver ID

License No

Save

Update

Cancel

Design and build a small system to calculate vehicle hire amounts

```
private void button3_Click(object sender, EventArgs e)
{
    con = new OleDbConnection("Provider=Microsoft.ACE.Oledb.12.0;Data Source = ayubo.mdb");
    con.Open();
    string query = "SELECT * FROM Vehicle where vehicle_id = @vehicle_id";
    cmd = new OleDbCommand(query, con);
    String vehicle_id = comboBox1.Text;
    cmd.Parameters.AddWithValue("vehicle_id", comboBox1.Text);

    dr = cmd.ExecuteReader();
    int weekly_rent = 0;
    int monthly_rent = 0;
    int daily_rent = 0;

    while (dr.Read())
    {
        weekly_rent = Int32.Parse(dr["monthly_rent"].ToString());
        monthly_rent = Int32.Parse(dr["weekly_rent"].ToString());
        daily_rent = Int32.Parse(dr["daily_rent"].ToString());
    }

    DateTime start_date = dateTimePicker1.Value;
    DateTime end_date = dateTimePicker2.Value;
    TimeSpan timeDifference = end_date - start_date;
    int differenceInDays = timeDifference.Days;
    int noMonths = differenceInDays / 30;
    int remainingDays = differenceInDays % 30;
    int noofweeks = remainingDays / 7;
    int noofdays = (differenceInDays - ((noMonths * 30) + (noofweeks * 7)));
    int total_bill = (noMonths * monthly_rent) + (noofweeks * weekly_rent) +
        (noofdays * daily_rent);

    textBox3.Text = total_bill.ToString();
}
}
```

Figure 14 : Rent a Vehical

Records of Customer Billing

```
private void button1_Click(object sender, EventArgs e)
{
    con = new OleDbConnection("Provider=Microsoft.ACE.Oledb.12.0;Data Source = ayubo.mdb");
    con.Open();
    string query = "insert into Rent (start_date,end_date,total_bill,vehicle_id,customer_id,driver_id) values (@start_date,@end_date,@total_bill,@vehicle_id,@customer_id,@driver_id)";
    cmd = new OleDbCommand(query, con);

    cmd.Parameters.AddWithValue("start_date", dateTimePicker1.Text);
    cmd.Parameters.AddWithValue("end_date", dateTimePicker2.Text);
    cmd.Parameters.AddWithValue("total_bill", textBox3.Text);
    cmd.Parameters.AddWithValue("vehicle_id", comboBox1.Text);
    cmd.Parameters.AddWithValue("customer_id", comboBox2.Text);
    cmd.Parameters.AddWithValue("driver_id", comboBox3.Text);
    cmd.ExecuteNonQuery();
    con.Close();
    MessageBox.Show("Rent bill has been added successfully.");
}
}
```

Debugging an Application

The process of finding and fixing flaws, sometimes known as "bugs," in software code that could lead to unexpected behavior or a crash is known as debugging. Debugging is the process of locating and fixing errors or faults in software or systems to stop them from operating incorrectly. Debugging gets more difficult when subsystems or modules are closely connected since changes made to one module may result in additional bugs popping up in other modules. Occasionally, debugging a program requires more time than coding it.

In order to debug a software, a user must first identify the issue, isolate its source code, and then resolve it. A program user has to understand how to resolve issues as well as problem knowledge.

Types of Debugging

1. **Determine the Error:** Poor error detection can result in lost development time. It is common for consumers to report production issues, which can be difficult to understand, and occasionally the information we get is inaccurate. It is critical to determine the true error.
2. **Locate the Error:** Following accurate error identification, you must search the code for the precise location of the error. At this point, locating the issue should take precedence than understanding it.
3. **Analyze the Error:** The third stage requires you to examine the code using a bottom-up methodology starting at the error location. This aids in your comprehension of the mistake. There are two primary objectives for bug analysis:
4. **Validate the Analysis:** After completing the analysis of the initial defect, you must identify a few additional potential issues with the application. Using a test framework, write automated tests for these sections in this stage.

5. Cover Lateral Damage: All of the unit tests for the code that you plan to modify must be created or gathered at this point. Now, all of these unit tests ought to pass if you run them.
6. Fix & Validate: This is the last step, where all issues are fixed and test scripts are executed to see whether they all pass.

Coding Standards

Coding standards are a set of rules and recommendations for creating code in a consistent, readable, and maintainable way. They are often referred to as coding conventions or programming style guidelines. These guidelines facilitate better communication among developers, increase comprehension of one another's code, and reduce error-proneness and manageability of the codebase. While coding guidelines could change between businesses and computer languages, the following universal components are typically highlighted:

Naming conventions: Specifies how variables, functions, classes, and other code elements should be named in order to make it easier to comprehend and maintain. These conventions frequently specify which identifier types should use camelCase, PascalCase, or snake_case.

Formatting and indentation: Using spacing, indentation, and formatting consistently makes the code easier to read. This covers guidelines for using spaces or tabs.

Code structure and organization: Arrangement best practices for logically and conveniently arranging code files, modules, and folders within a codebase. This covers conventions for class and function placement within files, code separation, and file management.

Error handling and exception management: Using standard procedures to deal with exceptions, errors, and edge cases can help to guarantee that the code responds gracefully and reliably in a variety of situations.

Guidelines for successfully utilizing functions, classes, and libraries to promote code reuse and modularity. This contains suggestions for reducing code duplication and producing reusable components.

Performance considerations: Best practices for maximizing the efficiency of code, such as recommendations for resource consumption, memory management, and algorithm complexity

References

- justinmind.com. (n.d.). Retrieved from <https://www.justinmind.com/prototyping/7stupidly-simple-login-forms-and-how-to-prototype-them-with-justinmind>
- medium.com. (n.d.). Retrieved from <https://medium.com/growininsights/why-use-codingstandards-b308c2c1aafa>
- simplicable.com. (n.d.). Retrieved from <https://simplicable.com/new/algorithm-vscode#:~:text=An%20algorithm%20is%20a%20series,steps%20that%20machines%20can%20execute.>
- techopedia.com. (n.d.). Retrieved from <https://www.techopedia.com/definition/5478/texteditor>
- zoomtute.com. (n.d.). Retrieved from <https://www.zoomtute.com/content/blogs/8/visualstudio-is-an-integrated-development-environment-dot>

Grading Rubric

Grading Criteria	Achieved	Feedback
LO1 Define basic algorithms to carry out an operation and outline the process of programming an application.		
P1 Provide a definition of what an algorithm is and outline the process in building an application.		
M1 Determine the steps taken from writing code to execution.		
D1 Evaluate the implementation of an algorithm in a suitable language. Evaluate the relationship between the written algorithm and the code variant		
LO2 Explain the characteristics of procedural, objectorientated and event-driven programming, conduct an analysis of a suitable Integrated Development Environment (IDE)		

P2 Give explanations of what procedural, objectorientated, and eventdriven paradigms are; their characteristics and the relationship between them.		
M2 Compare and contrast the procedural, object orientated and event driven paradigms used in given source code of an application		
D2 Critically evaluate the source code of an application which implements the programming paradigms, in terms of the code structure and characteristics.		
LO3 Implement basic algorithms in code using an IDE.		
P3 Write a program that implements an algorithm using an IDE.		
M3 Use the IDE to manage the development process of the program.		

D3 Evaluate the use of an IDE for development of applications contrasted with not using an IDE.		
LO4 Determine the debugging process and explain the importance of a coding standard		
P4 Explain the debugging process and explain the debugging facilities available in the IDE.		
P5 Outline the coding standard you have used in your code.		
M4 Evaluate how the debugging process can be used to help develop more secure, robust applications.		
D4 Critically evaluate why a coding standard is necessary in a team as well as for the individual.		

