Computer Science Pre-Released Material

By Tinapat (Game) Limsila

Original Date of Submission: 01 May 2021

For Cambridge CIE Computer Science Pre-Released Assessment

Table of Content

Task 1 – File Storage

Task 1.1 - Pseudocode

Design a record structure using pseudocode to store data about his books.

He wants to store the following:

- unique code for each book (between 100 and 999)
- title of the book
- main author
- year of publication.

```
DECLARE BookCode : INTEGER 100 to 999
DECLARE BookTitle  : STRING
DECLARE BookAuthor : STRING
DECLARE BookPublication : INTEGER
```

Task 1.2 – 1.4 -- Python

- Task 1.2 - Write program code to:
  - create your record structure
  - create an array to store the records about the books
  - input data for 10 books from the user
  - store each book as a separate record in an array
  - output the data in each record with an appropriate message.
- Task 1.3 - The program needs to store the records in the array into a file. Write a subroutine to store the records in a serial file
- Task 1.4 - Write a procedure to read the records from a serial file and output them with an appropriate message.

Notes: change the operator from “and” to “or” both conditions cannot be true ever

```
# Task 1.2
def enter_code():
    global book_code
    book_code = input("Book unique code: ")
    if (int(book_code) < 100) or (int(book_code) > 999):
        print("invalid")
        enter_code()
def enter_title():
    global book_title
    book_title = input("Title of the book: ")

def enter_author():
    global book_author
    book_author = input("Main author: ")

def enter_year():
    global book_year
    book_year = input("Year of Publication: ")
    if (int(book_code) <= 0 and int(book_code) >= 9999):
        print("invalid")
```

Commented [GL1]: Task 1.2
User input the first 3 digit of the book code with a check if the input number is between 100 and 999 if not it will notify the user and

Commented [GL2]: Task 1.2
Input the title of the book

Commented [GL3]: Task 1.2
Input the title of the author

```
        enter_year()

# Task 1.3
with open("BookCollection.txt", "a") as file:
    array = []
    for i in range(10): # its 10 in the exams
        print("This is for book #" + str(i+1))
        print("format example: 101_Harry Potter_JK_1997")
        enter_code()
        enter_title()
        enter_author()
        enter_year()
        book = book_code + "_" + book_title + "_" + book_author + "_" + book_year + "\n"
        array.append(book)
        print(book + "is stored")
    file.writelines(array)


# Task 1.4
with open("BookCollection.txt", "r") as file:
    booklist = file.readlines()
    i = 1
    print(booklist)
    for book in booklist:
        current_book = book.split("_")
        print("This is book number " + str(i))
        print("Book code: " + current_book[0])
        print("Book name: " + current_book[1])
        print("Book author: " + current_book[2])
        print("Book year of publication: " + current_book[3])
        i = i + 1
```

Commented [GL4]: Task 1.2
User input the year and make sure the date is sensible, a book published in the year 10,000 is not sensible

Commented [GL5]: Opening the file BookCollection.txt as append so it doesn't overwrite the also existed books, which "w" would.
Also if the file doesn't exist it would create one.

Commented [GL6]: Task 1.2
• create an array to store the records about the books

Commented [GL7]: Task 1.2
• input data for 10 books from the user

Commented [GL8]: Task 1.2
• create your record structure

Commented [GL9]: Task 1.2
Append the book in a record structure form to the array
&
As we append a new string to the array, it is separate from the previous string that was appended

Commented [GL10]: Task 1.2
• output the data in each record with an appropriate message.

Commented [GL11]: Task 1.3
Writing the array of 10 books into a text file

Commented [GL12]: The use of 'i' is for

Commented [GL13]: Task 1.4
The code is reading each line of the text file and extract the information and display them with human readable text.

Task 1.5- 1.6 – Python

- Task 1.5 - Each book has a unique code. The unique code allows the book's details to be stored in a random file using a hashing algorithm.
  Develop an appropriate hashing algorithm
- Task 1.6 - Manually calculate the file location for several books using your hashing algorithm.

```
MOD_NUMBER = 20 # the algorithm
# Task 1.6
def write_hashing_algorithms(book_string):
    book_id = book_string[0:3]
    hashed_id = int(book_id) % MOD_NUMBER # Task 1.5 Hashing Algorithm
    while hash_table[hashed_id] != "\n":
        hashed_id = hashed_id + 1
    hash_table[hashed_id] = book_string # Task 1.6/1.8

def read_hashing_algorithms(book_id):
    found = False
    out_of_range = False
    hashed_id = int(book_id) % MOD_NUMBER
    while not(found) and not(out_of_range):
        if hashed_id < (len(hashed_list_from_file)):
            if (hashed_list_from_file[hashed_id])[0:3] != book_id:
                hashed_id = hashed_id + 1
            else:
                found = True
        else:
            out_of_range = True
    if found:
        print(hashed_list_from_file[hashed_id])
    else:
        print("Not in the hashed list")

hash_table = []
for i in range(MOD_NUMBER * 2):
    hash_table.append("\n")

with open("BookCollection.txt", "r") as file:
    book_list = file.readlines()
    for book in book_list:
        write_hashing_algorithms(book)

with open("HashedCollection.txt", "w") as hashed_file:
    hashed_file.writelines(hash_table)

with open("HashedCollection.txt", "r") as hashed_file:
    hashed_list_from_file = hashed_file.readlines()
    print(hashed_list_from_file)

search1 = input("Find: ")
read_hashing_algorithms(search1)
search1 = input("Find: ")
read_hashing_algorithms(search1)
```

Commented [GL14]: Task 1.5 & Task 1.7
This is the hashing algorithm by taking the remainder of the book_id divided by 20, the remainder would be the address of the book in an array

Commented [GL15]: V2
I changed the variable name from book_string to book_id. It would still work as code was only to extract the first 3 character but it just makes more sense to change.

Commented [GL16]: To make sure to stop the while loop which the index of the list is too large or the book is found.

Commented [GL17]: Serial Search after the random search doesn't find the book_string

Commented [GL18]: to keep going through the list (serial) if the book is not in the expected position.

Commented [GL19]: If found print book if not "not found"

Commented [GL20]: This is for initialise the list as on python list of empty string need to be created manually
I timed the MOD_NUMBER by 2 to expand the array as if the remainder of the book code are all at the end of the list, I would have a room to overflow to.
The other method would've been loop back to the starting position but that would interfere the record random access.

Commented [GL21]: ***
This involves the previous program to be run first to generate the BookCollection.txt

Commented [GL22]: Task 1.6
The `write_hashing_algorithms()` `function calculate the book file location using the hashing algorithm, an the address correspond to the position in the array`

Commented [GL23]: Task 1.8
This section stored the hashed location into a text file, by write a new line using

Commented [GL24]: Task 1.9
This section read the hashed location from a text file.

Commented [GL25]: User input for searching a book by its book code.

Task 1.7 – Pseudocode

Write a pseudocode algorithm to perform the hash calculation

```
FUNCTION HashCalculator(BookCode : INTEGER) RETURN HashKey
      ModCoefficient ← 20
HashKey ← MOD(BookCode, ModCoefficient)
ENDFUNCTION
```

Commented [GL26]: The Input is the BookCode(which can be extracted from the BookString by using another function) and output a hashed version of the BookCode

Commented [GL27]:

Commented [GL28]: Dividing BookCode with ModCoefficient(20) and the remainder will be the HashKey, which will be use as an position of the array

Task 1.8 – Pseudocode

Write a pseudocode algorithm to store a record in its hashed location in the random file

```
PROCEDURE StoreInHashedLocation(HashKey : INTEGER, HashTable : ARRAY, BookCode :
BookTitle, BookTitle : STRING, BookAuthor : STRING, BookPublication : INTEGER)
      OPENFILE RandomFile.txt FOR READ
      READFILE RandomFile.txt, HashTable
      OPENFILE RandomFile.txt FOR WRITE
      StoredFlag ← FALSE
      Book ← “”
      Book.append(BookCode, BookTitle, BookAuthor, BookPublication)
      WHILE StoredFlag = FALSE
            IF (HashTable[HashKey] = “”) or (HashTable[HashKey] is empty)
                  THEN
                        HashTable[HashKey] ← Book
                        WRTIEFILE RandomFile.txt, HashTable
                        StoredFlag ← TRUE
                  ELSE
                        HashKey ← HashKey + 1
            ENDIF
      ENDWHILE
      CLOSEFILE RandomFile.txt
ENDPROCEDURE
```

Commented [GL29]: From the HashCalculator function

Commented [GL30]: An array initial consist of only empty string.

Commented [GL31]: Assumption is that the record layout has not been created yet

Commented [GL32]: Read the file to a variable HastTable

Commented [GL33]: Assign

Commented [GL34]: concatenating all book parameter into one string

Commented [GL35]:

Commented [GL36]: ELSE is only if the HashKey position is occupied then increase the index by 1 to the next position in the array until it is empty.

Task 1.9 – Pseudocode

Write a pseudocode algorithm to read a record from its hashed location in the random file

```
FUNCTION ReadInHashedLocation(HashTable : ARRAY, HashKey : INTEGER, BookCode : INTEGER)
RETURN BookString
      OPENFILE RandomFile.txt FOR READ
      READFILE RandomFile.txt, HashTable
      FoundFlag ← FALSE
      WHILE FoundFlag = FALSE
            IF MID(HashTable[HashKey], 1, 3) = BookCode
                  THEN
                        BookString ← HashTable[Hashkey]
                        FoundFlag ← TRUE
                  ELSE
                        HashKey ← HashKey + 1
            ENDIF
      ENDWHILE
ENDFUNCTION
```

Commented [GL37]: Again, assuming that as input is the function that calculate the HashKey from the BookCode

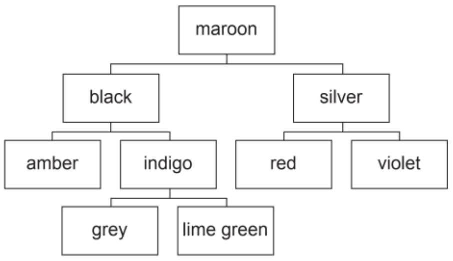
Commented [GL38]: Import string as an array

Commented [GL39]:

Task 2 – Binary Tree

Task 2.1

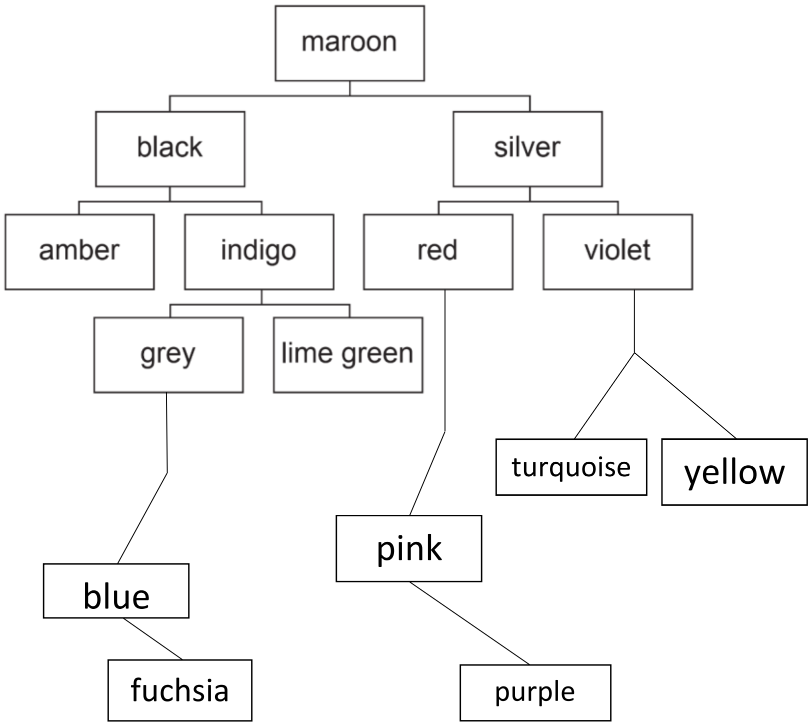
Identify the root node and leaf nodes in the binary tree.



```
Root node is maroon
Leaf node are black, amber, indigo, grey, lime green, sliver, red, violet
```

Task 2.2

Add the following data to the binary tree: **pink, yellow, blue, purple, fuchsia, turquoise**



Task 2.3 – Task 2.6 - Python

- Task 2.3 - Write program code to store the binary tree as a 1D array of records.
- Task 2.4 - Write program code to add a new data item to the binary tree
- Task 2.5 - Write program code to find the position of a specific colour in the binary tree
- Task 2.6 - Write program code to output the contents of the binary tree in alphabetical order

```
duplicate_flag = False
duplicate_list = []

class Node():
    def __init__(self, colour, position = None, left_pointer = None, right_pointer = None)
:
        self.position = position
        self.colour = colour
        self.left_pointer = left_pointer
        self.right_pointer = right_pointer

Maroon = Node('maroon', 0, 1, 2)
Black = Node('black', 1, 3, 4)
Silver = Node('silver', 2, 5, 6)
Amber = Node('amber', 3)
Indigo = Node('indigo', 4, 7, 8)
Red = Node('red', 5)
Violet = Node('violet', 6)
Grey = Node('grey', 7 )
Lime_Green = Node('lime green', 8)
#Task 2.3
binarytree = [Maroon, Black, Silver, Amber, Indigo, Red, Violet, Grey, Lime_Green]

def print_node(Node):
    print('''
Node Position {}
Node Colour: {}
Node Left Pointer: {}
Node Right Pointer: {}'''.format(Node.position, Node.colour, Node.left_pointer, Node.right
_pointer))

def asciify(Node, char_position = 0):
    return ord(Node.colour[char_position])
# Task 2.4
def adding_function(Append_Node, Parent_Node = binarytree[0], char_position = 0):
    global duplicate_flag
    if (asciify(Append_Node, char_position)) > (asciify(Parent_Node, char_position)):
        if Parent_Node.right_pointer == None:
            Append_Node.position = len(binarytree)
            Parent_Node.right_pointer = len(binarytree)
            binarytree.append(Append_Node)
        else:
            adding_function(Append_Node, binarytree[Parent_Node.right_pointer])
    elif (asciify(Append_Node, char_position)) < (asciify(Parent_Node, char_position)):
        if Parent_Node.left_pointer == None:
            Append_Node.position = len(binarytree)
```

Commented [GL40]: Task 2.4
New data item being Node
I created a class of object call nodes which consist of colour that we are storing, position of the node on the list, left & right pointer

Commented [GL41]: Task 2.1
Storing them in a 1D list called binarytree, consist of object

Commented [GL42]: For displaying the parameter of each node in human readable form

Commented [GL43]: For returning the first(Default, unless different parameter given) character of colour as an ASCII value to compare later in the adding function

Commented [GL44]:

Commented [GL45]: Base case

Commented [GL46]: If the append node is larger than the parent node then we recuse and pass the right node as the parent node and do the comparison again

Commented [GL47]: Base case

```
            Parent_Node.left_pointer = len(binarytree)
            binarytree.append(Append_Node)
        else:
            adding_function(Append_Node, binarytree[Parent_Node.left_pointer])
    else:
        if Append_Node.colour == Parent_Node.colour:
            duplicate_flag = True
            duplicate_list.append(Append_Node.colour)
        else:
            adding_function(Append_Node, Parent_Node, char_position +1)
# Task 2.5
def find_function(Wanted_Node, Parent_Node = binarytree[0], char_position = 0):
    if (asciify(Wanted_Node, char_position)) > (asciify(Parent_Node, char_position)):
        find_function(Wanted_Node, binarytree[Parent_Node.right_pointer], char_position)
    elif (asciify(Wanted_Node, char_position)) < (asciify(Parent_Node, char_position)):
        find_function(Wanted_Node, binarytree[Parent_Node.left_pointer], char_position)
    else: # when they equal
        if Wanted_Node.colour == Parent_Node.colour:
            print("Found {} at position {} of the list(Starting at 0)".format(Parent_Node.
colour, Parent_Node.position))
        else:
            find_function(Wanted_Node, Parent_Node, char_position +1)

def print_in_alphabetical_order(Parent_Node = binarytree[0]):
    if ((Parent_Node.left_pointer == None) and (Parent_Node.right_pointer == None)):
        print(Parent_Node.colour)
    if Parent_Node.left_pointer != None:
        print_in_alphabetical_order(binarytree[Parent_Node.left_pointer])
    if Parent_Node.right_pointer != None:
        print(Parent_Node.colour)
        print_in_alphabetical_order(binarytree[Parent_Node.right_pointer])

def __main__():
    Pink = Node("pink")
    adding_function(Pink)

    Yellow = Node("yellow")
    adding_function(Yellow)

    Blue = Node("blue")
    adding_function(Blue)

    Purple = Node("purple")
    adding_function(Purple)

    Fuchsia = Node("fuchsia")
    adding_function(Fuchsia)

    Turquoise = Node("turquoise")
    adding_function(Turquoise)

    Turquoise = Node("turquoise")
    adding_function(Turquoise)
```

Commented [GL48]: Else is only when the character are the same then compare the next character

Commented [GL49]: Task 2.5
For finding the position of the

Commented [GL50]: If the colour match then we found the position as the position is an attribute of the node, as its as simple as print the node.position

Commented [GL51]: Print an output of where the position of the colour is at in the 1D list

Commented [GL52]: Traverse through the binary tree from the bottom left to the bottom right, and due to the tree already being sorted when imputed, no sorting is required afterward

Commented [GL53]: Task 2.2
Adding new nodes to the Binary tree

Commented [GL54]: Show that duplication detection works

```
    for node in binarytree:
        print_node(node)
    if duplicate_flag:
        print()
        print("""all of these colors are already on the list
Duplicated not added""")
        print(duplicate_list)
    find_function(Yellow)
    print_in_alphabetical_order()

__main__()
```

Commented [GL55]: For all the nodes in the 1D array print all the parameters of the node in English.
Function print node() consist of printing all the nodes parameters: colour, position, left pointer, right pointer

Commented [GL56]: If the user enters a duplicate colour a message will be posted

Commented [GL57]: Task 2.5
Find position of colour yellow

Commented [GL58]: Task 2.6
Output all colours in the binary tree in alphabetical order

Commented [GL59]: Run Program

Task 3 – Object-oriented programming

Task 3.1 – 3.5 - Python

- Task 3.1 – The program needs a class for the tools.

  The information stored about each tool must include:

  - Name
  - Cost
  - Image file name (e.g. 'spade.jpg')
- Task 3.2 - Write program code to create a get and set method for each of the tool attributes
- Task 3.3 - The program needs a class for the shelves in the store.

  Each shelf has the following information. position of the shelf on the wall (between 0 and 4)

  array of objects of type tool (maximum of 10 items per shelf).

  Write program code for the class shelves.
  The constructor should set the position of the shelf but should not set any tools.
  Write program code for the attributes and the constructor method.
- Task 3.4 - Write program code for a set method to add a new tool to the shelf in the next available position.
- Task 3.5 - Write program code to define a procedure that takes a shelf object as a parameter and outputs the name and cost of each tool on that shelf

```
import random
class Tools: # Task 3.2
    def __init__(self, name, cost, image_file_name):
        self.name = name # string
        self.cost = cost # reals
        self.image_file_name = image_file_name # string

class Shelves:
    def __init__(self, array):
        self.position = random.randint(0,4) # only 0-4
        self.array = array

# Task 3.4
def add_tool_to_shelf(tool, shelf):
    if len(shelf.array) < 10:
        shelf.array.append(tool)
    else:
        print("shelf is full, max 10 tools")

def read_tool_from_shelf(shelf):
    for i in range(len(shelf.array)):
        current_tool = Shelf1.array(i)
        print(i)
        print(current_tool)

# Task 3.5
def list_item_in_self(shelf):
    i = 1
    for tool in shelf.array:
        print("Tool number {0}".format(i))
```

Commented [GL60]: Task 3.1
Created a class for tools
The attributes
- Name
- Cost
- Image File name

It takes the parameter from the function input.

Commented [GL61]: Task 3.3
A shelf class
The constructor should set the position of the shelf, hence the randint for generating random position for the shelf.
Array is the tool, initially its passed an empty array

Commented [GL62]: Task 3.4
When a tool as an input is passed to the function, check if the shelf if full (= 10) if not full then add tool to the shelf. If the shelf is full then output that its full and not add the tool.

Commented [GL63]: For item in the shelf.

```
        print("Tool: {}".format(tool.name))
        print("Cost: ${}".format(tool.cost))
        print()
        i += 1

Object1 = Tools("Hammer", "100", "hammer.jpg")
Object2 = Tools("Premium Hammer", "200", "hammer.jpg")

BASE_ARRAY = []
Shelf1 = Shelves(BASE_ARRAY)
add_tool_to_shelf(Object1, Shelf1)
add_tool_to_shelf(Object2, Shelf1)
add_tool_to_shelf(Object1, Shelf1)
add_tool_to_shelf(Object2, Shelf1)
add_tool_to_shelf(Object1, Shelf1)
add_tool_to_shelf(Object2, Shelf1)
add_tool_to_shelf(Object1, Shelf1)
add_tool_to_shelf(Object2, Shelf1)
add_tool_to_shelf(Object1, Shelf1)
add_tool_to_shelf(Object2, Shelf1) # 10th
add_tool_to_shelf(Object1, Shelf1) # 11th
input("Tool added, enter to list all item")
list_item_in_self(Shelf1)
```

Commented [GL64]: Adding words to make sense of the output parameters

Commented [GL65]: Task 3.5
This function takes the array from the shelf and print out the information of the tool in the shelf, with appropriate output

Commented [GL66]: Task 3.2
Get and set method for each tool in one line
The first parameter is the name of the tool, second is the cost in integer, three is the name is the file

Commented [GL67]: Array is the tool, initially its passed an empty array(BASE_ARRAY)

Commented [GL68]: Adding new tools to the list, 11 tools in total to see ensure that if the shelf is full the a proper message is output instead of an error.

Commented [GL69]: