# AS Computer Science in class quiz

Name:_____Marks: _____.

Q1

A programming language has the built-in function CONCAT defined as follows:

```
CONCAT(String1 : STRING, String2 : STRING [, String3 : STRING] )  RETURNS STRING
For example:  CONCAT("San", "Francisco") returns "SanFrancisco"
              CONCAT("New", "York", "City") returns "NewYorkCity"
```

The use of the square brackets indicates that the parameter is optional.

**(a)** State the value returned by the following expressions.

If the expression is not properly formed, write ERROR.

**(i)** `CONCAT("Studio", 54)` ..... [1]

**(ii)** `CONCAT("parity", "error", "check")` ..... [1]

**(iii)** `CONCAT(CONCAT("Binary", "▼", "Coded"), "▼", "Decimal")`

▼ indicates a <Space> character

.....[2]

Q2

A string encryption function is needed. The encryption uses a simple character-substitution method.

In this method, a new character substitutes for each character in the original string. This will create the encrypted string.

The substitution uses the 7-bit ASCII value for each character. This value is used as an index for a 1D array, `Lookup`, which contains the substitute characters.
`Lookup` contains an entry for each of the ASCII characters. It may be assumed that the original string and the substitute characters are all printable.

For example:

- 'A' has ASCII value 65
- Array element with index 65 contains the character 'Y' (the substitute character)
- Therefore, 'Y' substitutes for 'A'
- There is a different substitute character for every ASCII value

The programmer writes a function, `EncryptString`, to return the encrypted string. This function will receive two parameters, the original, `PlainText` string and the 1D array.

**(a)** The first attempt at writing the pseudocode for this function is shown below.

Complete the pseudocode.

For the built-in functions list, refer to the **Appendix** on page 4

```
FUNCTION EncryptString(.....) RETURNS STRING
    DECLARE ..... , ..... : CHAR
    DECLARE OldCharValue : .....
    DECLARE n : INTEGER
    DECLARE OutString : STRING

    ..... //initialise the return string

    //loop through PlainText to produce OutString
    FOR n ← 1 TO ..... //from first to last character
       OldChar ← .....//get next character
       OldCharValue ← .....//find the ASCII value
       NewChar ← .....//look up substitute character
       .....//concatenate to OutString
    ENDFOR

    .....
ENDFUNCTION
```

[10]

**(b)** Additional code needs to be written to allow the user to change some of the characters in the array `Lookup`.

The user will input:

- the array start position
- the number of elements to change
- each new substitute character

At the end, the program will finally output a confirmation message.

The first version of the algorithm is represented by the flowchart on the following page.

**(i)** Write **program code** to declare the array `Lookup`.

Programming language .....

.....

.....

.....[2]

**(ii)** Write **program code** to implement the flowchart design.

In addition to the `Lookup` array, assume that the following variables have been declared:

```
StartPos, NumToChange, n, NewChar
```

Programming language .....

.....

.....

.....

.....

.....

.....

.....

.....

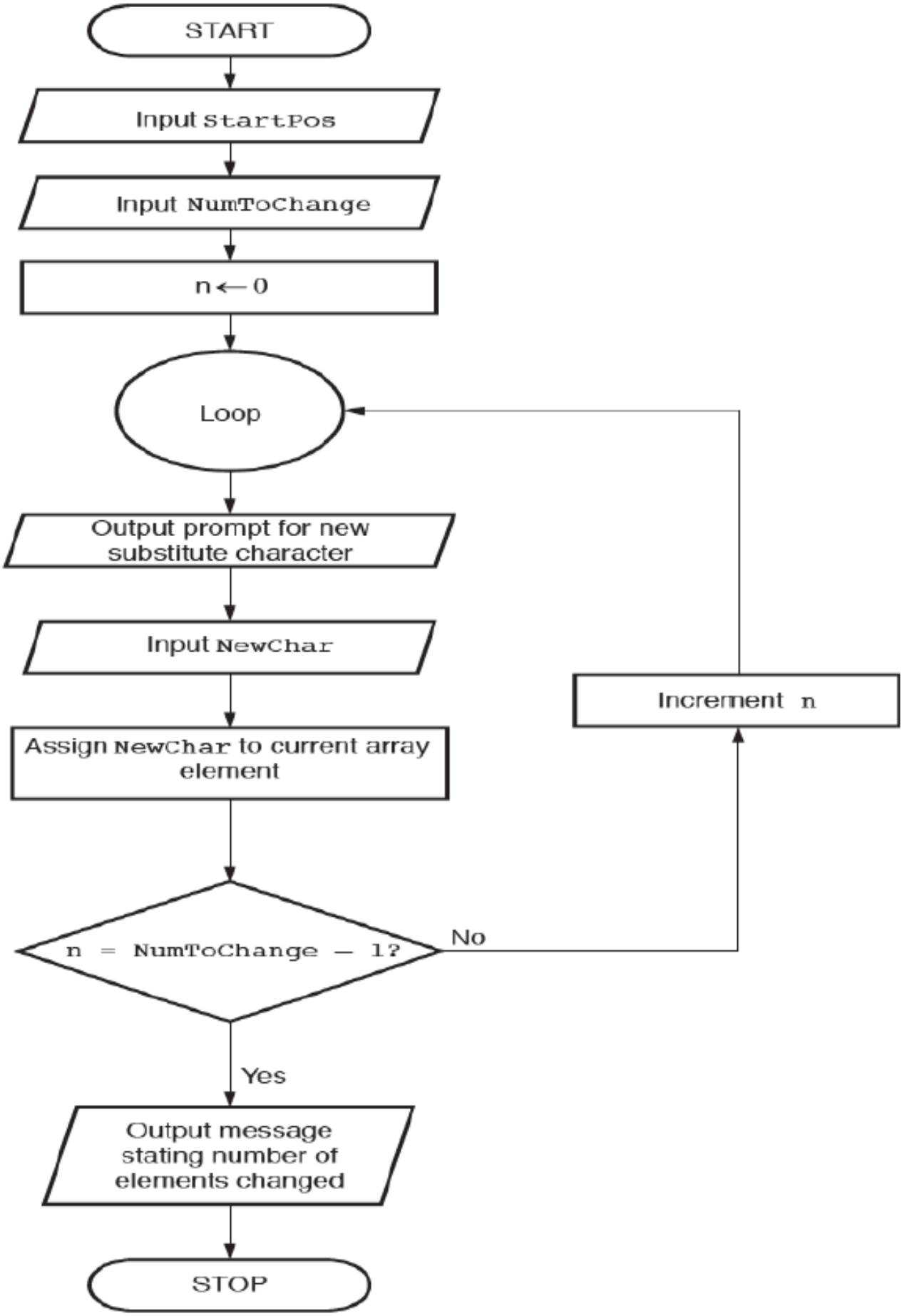
.....

.....

.....

.....

.....[6]


START
Input StartPos
Input NumToChange
n ← 0
Loop
Output prompt for new substitute character
Input NewChar
Assign NewChar to current array element
n = NumToChange – 1?
No
Increment n
Yes
Output message stating number of elements changed
STOP

Appendix

```
import random
# String manipulation functions:
str1 = "A Level Computer Science Course"
print (str1[0]) # print the first character in the given string
print (str1[1]) # print the second character in the given string
### ....and so on until the end of the string
print (len(str1)) # prints out the length of the string

# Slicing in Python
print (str1[3:8]) # from position 3 to 8 not including 8
print (str1[:2]) # all characters to the end of the string
print (str1[-2:]) # all characters from the beginning of the string
print (str1[2:]) # slice from end of the string
print (str1[:-2]) # terminates the string at that position

# Concatenation of strings
print (str1 + " " + "is very difficult for me!") # use of + operator

# ASCII codes
print (ord('a')) # gives out the ASCII value of 'a' according to the ASCII table

# You can check for values for any characters using ord() function....
print (chr(97)) # gives out the character for the given value according to ASCII table

# Random number generator
print (random.randint(1, 6)) # we need to import random module to use this function
```

**(a)** — **10**

```
FUNCTION EncryptString (LookUp : ARRAY, PlainText : STRING)
RETURNS STRING

DECLARE OldChar, NewChar : CHAR
DECLARE OldCharValue : INTEGER
DECLARE OutString: STRING

//first initialise the return string

OutString ← ""   //initialise the return string

//loop through PlainText to produce OutString

FOR n ← 1 to LENGTH(PlainText)   //from first to last character
OldChar ← MID(PlainText, n, 1)   //get next character
OldCharValue ← ASC(OldChar) //find the ASCII value
NewChar ← Lookup[OldCharValue]   //look up substitute character
OutString ← Outstring & NewChar  // concatenate to OutString

ENDFOR

RETURN OutString // EncryptString ← OutString

ENDFUNCTION
```

One mark for each part-statement (shown underlined and bold)

**(b) (i)** — **2**

**VB**: `Dim Lookup(0 to 127 / 128) As CHAR`

**Pascal**: `Var Lookup: Array[0..127 / 1..128] Of CHAR`

**Python**:
```
Lookup = ["" for i in range(128)]
```

*OR*

```
Lookup = []
For i in range(128) :
Lookup.append("")
```

Mark as follows:
VB / Pascal: one mark per part-statement as underlined and bold
Python:
One mark for `Lookup` **and** `[]`
One mark for `range(128)`

(ii) 'Pseudocode' solution included here for development and clarification of mark scheme.
Programming language solutions appear in the Appendix. **6**

```
INPUT StartPos                                          (1)
INPUT NumToChange

FOR n ← 0 to NumToChange - 1                            (2)
   OUTPUT " Input new value for position "              (3)
   INPUT NewChar                                        (4)
   Lookup[StartPos + n] ← NewChar                       (5)
ENDFOR

OUPUT (NumToChange & " entries changed")                (6)
```

**ALTERNATIVE:**

```
INPUT StartPos                                          (1)
INPUT NumToChange
n ← 0

REPEAT                                                  (2)
   OUTPUT " Input new value for position "              (3)
   INPUT NewChar                                        (4)
   Lookup[StartPos + n] ← NewChar                       (5)
   n ← n + 1
UNTIL n = NumToChange

OUTPUT (NumToChange & " entries changed")               (6)
```

Mark points as circled, descriptions as below:

1. Two `INPUT` statements
2. Working loop (using values of n from flowchart)
3. `OUTPUT` prompt (exact text not specified)
4. `INPUT NewChar`
5. Assignment of `NewChar` to correct array element
6. `OUTPUT` final message after loop (exact text not specified but must include `NumToChange` or loop counter if value correct at that point)

```
StartPos = int(input("Enter start position: "))
NumToChange = int(input("Enter number to change: "))
for n in range(NumToChange) :
      NewChar = input("Input new value for position: ")
      LookUp[StartPos + n - 1] = NewChar
print(str(NumToChange) + " entries changed")
```

**ALTERNATIVE:**

```
StartPos = int(input("Enter start position: "))
NumToChange = int(input("Enter number to change: "))
n = 0
while n < NumToChange :
      NewChar = input("Input new value for position: ")
      LookUp[StartPos + n] = NewChar
      n = n + 1
print(str(NumToChange) + " entries changed")
```