South West Regional Mock
2023 Moderation
0795 Computer Science Marking Guide

PAPER 1:

1	C
2	B
3	D
4	B
5	D
6	A
7	A
8	C
9	A
10	D

11	B
12	B
13	B
14	C
15	C
16	C
17	C
18	D
19	A
20	D

21	A
22	B
23	B
24	B
25	D
26	C
27	D
28	B
29	A
30	C

31	B
32	C
33	B
34	C
35	A
36	B
37	B
38	D
39	A
40	D

41	C
42	B
43	A
44	C
45	C
46	C
47	A
48	B
49	D
50	A

PAPER 2:

1. (i) (a)

128	64	32	16	8	4	2	1
1	0	1	1	0	1	1	0

$128 + 32 + 16 + 4 + 2 = 182_{10}$

(b) -128 to 127 $(2^7$ to $2^7 - 1)$

(c)

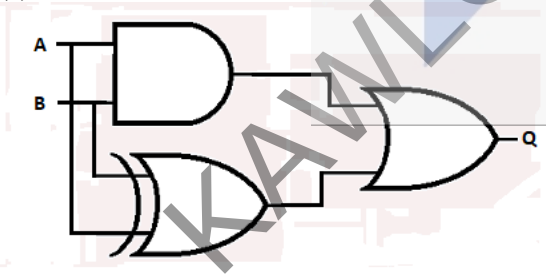
1011	0110
B	5

Answer: $10110110 = B5_{16}$

(d)- Hex numbers take up less screen space.
- Hex numbers are easier and faster to read and work with
- Mistakes are less likely and easier to trace/debug.

(ii) (a) $B \cdot (\overline{\overline{A} + \overline{B}}) = B.(\overline{\overline{A}} \cdot \overline{\overline{B}}) = B \cdot A \cdot B = A \cdot B$

(b)



(c) Reduced number of gates:
- reduces power consumption
- reduces the heat generated by the chip
- increases the speed

---

2. (i) (a) The size of the data bus is 32 bits

The number of address line is $2^9 x 2^{20} = 2^{29}$ = 29 address lines

(b) Number of chips = $\frac{4096 \times 16}{128 \times 8} = 64$

(c) Cache memory is high-speed memory between the CPU and main memory. Cache memory is different from main memory in that cache is static RAM (does not require constant refresh) while main memory is dynamic RAM (needs periodic refresh).

(ii) (a) Load/store architecture means that:

- Memory operands can only be accessed using load and store operations
- All operands involved in arithmetic or logic operations must either be in processor registers or one of the operands may be given explicitly in the instruction.

(b) *Advantages of RISC processors over CISC*

- Higher performance since simplified instruction sets allow for a pipeline processing.
- Lower per chip cost as a simple instruction set uses up much less chip space
- Shorter design cycles allow them to take advantage of other technological developments sooner than CISC processors

(c)
```
LDR R1, A                  //R1 ← [A]
LDR R2, B                  //R2 ← [B]
ADD R3, R1, R2             //R3 ←[R1] + [R2]
STR R3, C                  //C ← R3
```

---

3. (i) (a) A scheduling policy is an algorithm that is used to determine which process in the ready state should be moved to the running state (allocated the CPU)

(b) Throughput is the number of processes that are completed per unit time. Throughput has to be maximal as possible for a good scheduling policy.

Turnaround time is the interval from the time of submission of a process to the time of completion. A good scheduling policy should minimize turnaround time as possible.

(c) Convoy effect is the slowdown of a system resulting from the fact that many short running (I/O bound) processes are waiting for one or few long running (CPU-bound processes) to get off a resource they need.

FCFS scheduling may lead to convoy effect.

(ii)(a)-Allocating and de-allocating memory space as needed

-Keeping track of which parts of memory are currently being used

-Deciding which processes are to be loaded into memory when memory space becomes available

(b)Virtual address space is the set of all the logical addresses generated about a program by the CPU.

Physical address space is the set of all physical addresses mapped to the logical addresses.

(c)The parts (pages) of both programs that immediately needed are loaded into main memory while those not needed immediately are kept in virtual memory

When a page is referenced and it is in virtual memory, it is swapped in while a page in memory is swapped out.

This constant swapping in an out of pages allows both programs to run simultaneously even though both programs are too large to fit into the computer's memory

(d) Advantages of dynamic partitioning:

- No internal fragmentation
- No limitation on the size of a process
- Degree of multiprogramming is dynamic

---

4. (i) (a) A record is a collection of fields (attributes) that describe an entity.

(b) Random file organization.

Advantages:

- There is fast access to records
- It is easy to update files
- It does not require the use of indexes so it saves space
- Transactions do not need to be sorted before being updated

(c)

CustomerID	RecordKey
1081	57
1239	215
1999	975

(ii) (a)

```
CREATETABLEStudents(
  studentId     int
  studentName   varchar
  DOB  date/time
  PrimaryKey (studentId)
);
```

```
CREATETABLESubjects(
  subjectCode     int
  subjectTitle    varchar
  PrimaryKey (subjectCode)
  );
```

```
CREATETABLEResults(
  studentId       int
  subjectCode     int
  grade      char
  PrimaryKey (studentId, subjectCode)
  Foreign Key References Students(studentId)
  Foreign Key ReferencesSubjects(subjectCode)
);
```

(b) Referential integrity is a database constraint that requires that each foreign key value must match a primary key value. This ensures that the relationship between tables in a database remains accurate.

(c)
```
Foreign Key ReferencesStudents(studentId)
```
Or,
```
Foreign Key ReferencesSubjects(subjectCode)
```

(d)

tblstudents

studentId	studentName	dateOfBirth
22ST001	Jane Beza	01/04/2005

tblSubjects

subjectCode	subjectTitle
0795	Computer Science

tblResults

studentId	subjectCode	grade
22ST001	0795	C

---

5. (i) (a) 18, 15, 25, 7

Pass 1	18	15	25	7
	15	18	25	7
	15	18	25	7
	15	**18**	**7**	**25**
Pass 2	15	18	7	25
	15	18	7	25
	15	**7**	**18**	**25**
Pass 3	15	7	18	25
	7	**15**	**18**	**25**

(b) Comparisons: 3 + 2 + 1 = **6**
Swaps: 1 + 1 + 1 = **3**

(c) For list of size n, we have

Total = (n-1) + (n-2) + (n-3) + … + 3 + 2 + 1 = n(n-1)/2

= $(n^2 + n)/2 = O(n^2)$

(ii) (a) Software testing is the process of executing software with the intent of finding errors or bugs.

(b)Testing is necessary because:

-It allows us to detect bugs in a software

-It allows us to check software's compliance with user requirements

-Faulty software could be life-threatening in a critical system

-It could be costly to right the damage caused by faulty software

(c)Normal data: valid data which the system will accept

Abnormal data: erroneous or invalid data which the system will reject

Extreme data: data values that are chosen at the absolute limits of the normal range

---

6. (i) (a) An IP address is a unique number that identifies a device on the Internet.

Components: Network ID and Host ID

(b) Class A: 0.x.x.x.x to 127.x.x.x.x

Class B: 128.x.x.x to 191.x.x.x.x

(ii)(a)Serial transmission is the transfer of several bits one after the other over the same line.

(b)Synchronous transmission uses a timing signal to ensure that the transmitter and the receiver are in step with one another.

Asynchronous transmission uses start and stop bits to specify the beginning and end of each character.

(c)The purpose of the parity bit is error detection.

Parity bit: **0**

(iii)(a)A router connects two or more dissimilar networks together and determines the best path that a message will take from one network to another.

(b) A bridge is used to connect two separate but similar local area networks or to divide one LAN into segments.

(c) A modem converts digital signals from a computer to analog signal for transmission over an analog link, and converts analog signals at reception back to digital.

---

7. (i) (a) A variable is named location that holds a value can change during the execution of the program.

Variable declaration consists of stating a variable's name and data type in a computer program.

(b) Variable declaration is important because:

- It tells the computer how much space to allocate in memory to hold the variable's value
- It tells the computer the operations that define possible manipulations to be performed on the variable
- It tells the computer how to interpret the bit patterns used to store the variables value
- It gives the variable a name that will be used during code generation

(c) Syntax error

(ii) (a)

While loop	Repeat loop
- Condition is evaluated before execution of the loop body (pre-test loop)	- Condition is evaluated after execution of the loop body (post-test loop)
- Loop body may never execute (if the condition is false)	- Loop body must execute at least once (even if the condition is false)

(b)

```
begin
  sum ← 0
  for(i ← 1 to 5)do
    Get number
    sum ← sum + number
  next i
  print sum
end
```

(c) A recursive function is one that calls itself.

Iteration is faster at runtime because:

- Each recursive call creates an entry(stack frame) at the top of a stack to store the return address from the function, as well as each call's parameters and local variables.

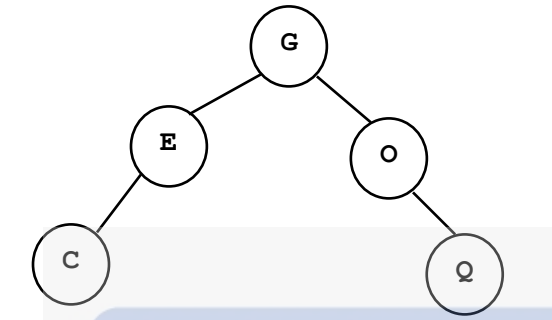
- This means that extra time is used (overhead) in maintaining the stack for each call.
- Iteration does not use a stack, so no time is wasted in maintaining a stack, making iteration faster.

---

8. (i) (a) A complete binary tree is a binary tree in which every level, except possibly the last, is completely filled and all nodes are as far left as possible.

(b) Pre-order: Node – Left – Right
Post-order: Left – Right – Node
Pre-order traversal: `L - E - C - G - O - Q`
Post-order traversal: `C - G - E - Q - O - L`

(c)



(d)

L	E	O	C	G	-	Q

(ii)

```
-   S2.push(S1.pop())     or x = S1.pop(), S2.push(x)
-   S2.push(S1.pop())        x = S1.pop(), S2.push(x)
-   S3.push(S1.pop())        x = S1.pop(), S3.push(x)
-   S3.push(S1.pop())        x = S1.pop(), S3.push(x)
-   S3.push(S2.pop())        x = S2.pop(), S3.push(x)
-   S2.push(S2.pop())        x = S2.pop(), S3.push(x)
```

PAPER THREE
SECTION A: DATABASE
Task 1: Normalisation

(i) ***Given***: `{StudentID, BookID} → {ReturnDate}`
***Expected*:**
`{StudentID} → {StudentName}`
`{StudentID} → {Gender}`
`{StudentID} → {DOB}`
`{StudentID} → {Class}`
`{BookID} → {BookTitle}`

Accept:
`{StudentID} → {StudentName,Gender,DOB,Class}`

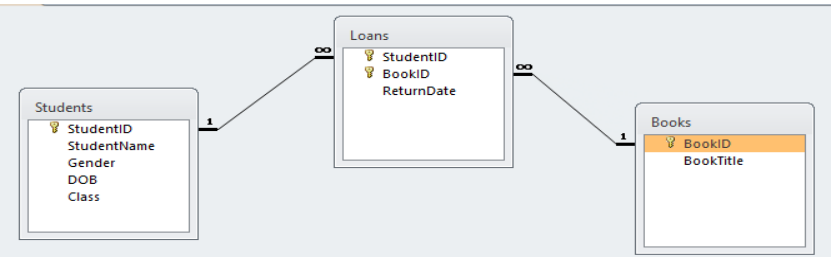
(ii)
```
Students(StudentID, StudentName, Gender, DOB, Class)
Books(BookID, BookTitle)
Loans(StudentID, BookID, ReturnDate)
```

Task 2: Implementation

(i) Evidence of Library database
Evidence of all three tables
SQL statement: **`CREATE DATABASE`** `Library;`

(ii)
```
CREATE TABLE Students(
  StudentID        char
  StudentName      varchar
  Gender     char(1)
  DOB        date/time
  Class      varchar
  PRIMARY KEY StudentID
);
```

(iii)



(iv) **Students**

Students

StudentID	StudentName	Gender	DOB	Class
21LS001	Alem Joseph	M	2/3/2006	LS2
21LA010	Abianga Moses	M	6/24/2005	LA4
22LA015	Bih Anita	F	4/22/2006	LA2
22LS020	Enanga Laura	F	8/29/2007	LS1

Books

Books

BookID	BookTitle
BK01	A/L Chemistry
BK02	A/L History
BK03	A/L Computer Science
BK04	A/L Mathematics

Loans

Loans | Books

StudentID	BookID	ReturnDate
21LS001	BK01	11/12/2022
21LA010	BK02	12/1/2022
21LA010	BK04	1/10/2023
22LA015	BK02	11/5/2022
22LS020	BK03	1/10/2023

(v)
```
SELECT Students.StudentID, Students.StudentName, COUNT(*)
FROM Students INNER JOIN Loans ON Students.StudentID = Loans.StudentID
GROUP BY Students.StudentID, Students.StudentName;
```
Or,
```
SELECT Students.StudentID, Students.StudentName, COUNT(*)
FROM Students, Loans
WHERE Students.StudentID = Loans.StudentID
GROUP BY Students.StudentID, Students.StudentName;
```

Task 3: Program Design

(i)

(ii)
```
Get num
count ← 0
while(num ≠ 0)do
  R[count] ← num mod 2
  num ← num div 2
  count ← count + 1
endwhile
```

(iii) (a) Base is valid when it is either 2 or 10.

```
for(i ← count-1 downto 0)do
  print(R[i])
next i
```

(b)
```
valid_base ← 0
do
  print("Enter a base:");
  get(base)
  if(base = 2 or base = 10)then
    valid_base ← 1
  else
    print("Invalid base.")
  endif
while(valid_base = 0)
```

Task 4: Implementation

(i)
```
void decBin(int deci){
  int R[15], count = 0;
  while(deci != 0)
  {
   R[count] = deci % 2;
   deci = deci / 2;
   count++;
  }
  printf("\nBinary: ");
  for(int i = count-1; i >= 0; i--)
    printf("%d",R[i]);
}
```

(ii)
```
void binDec(int bin){
  int rem, sum = 0, p = 0;
  while(bin != 0)
  {
    rem = bin % 10;
    bin = bin / 10;
    sum = sum + rem * pow(2,p);
    p = p + 1;
  }
  printf("\nDecimal: %d", sum);
}
```

(iii) (a)
```
do{
  printf("Enter a base: ");
  scanf("%d", &base);
  if(base == 2 || base == 10)
    valid_base = 1;
  else
    printf("Not a valid base.\n\n");
  }while(valid_base == 0);
```

(b)
```
print("Enter a base %d number: ", base);
scanf("%d", &num);
```

(c)
```
if(base == 2){
  int check = num;
  while(check != 0){
    if(check % 10 > 1){
      printf("Not a binary number!\n\n");
      goto L0;
    }
    else
      check = check / 10;
  }
  printf("\nBinary: %d", num);
  binDec(num);
```

```
    }
  else{
    printf("\nDecimal: %d", num);
    decBin(num);
  }
```

(iv) See (iii)(c) above

(vi)