IJC Prelim 2 Paper 1 Solution

1) a) i) 27 [1]

ii)
- Short piece of user friendly word or string to stand for the operation in a low level language instruction.
- Used to make program code more easy to remember. In this case ADD replaces a binary code

[2]

iii)
1. Address of instruction in Program counter(PC) copied to Memory Address Register(MAR)
2. Contents of address in MAR copied to Memory Data Register(MDR)
3. Contents of MDR copied to Current Instruction Register(CIR)
4. Instruction in CIR decoded
5. Instruction executed
6. Increment PC (at any stage between step 1 and 5)

[6]

b) i) Consists of
- Windows
- Icons
- Menus
- Pointer
- School/children/inexperienced users/ home computer (almost any application). Any one

[3]

ii)
- (Questions and) spaces for answers shown on screen/insertion boxes
- Input can be by radio buttons
- Pop up menus/drop down lists
- Insertion fields provided with validation checks
- Mirrors a hardcopy form

- Any example where on screen input is necessary (eg online survey form, membership application form etc)

[3]

c) Disk formatting
- To divide up the surface of a disk into more easily manageable sectors
- Copy editor will use a hard disk which will need to be formatted before being used/to store texts (2)

File compression.
- Reduces size of files without the loss of any detail.
- To speed up the transfer of files which are very large.

Virus Checker
- Used to check any files on or entering the system for viruses
- The staff will use the communications regularly and hence files will be subject to attack / manyfiles being received.

Hardware driver
- Used to control communication between computer and peripherals
- Controls formatting and fonts of text sent to the printer.

Any relevant 3 utilities.

[6]

2 a) The technique of taking the problem as a whole and breaking it into a small number of well defined, interconnected but separate sub-problems; Each of these sub-problems is then separately broken down into smaller sub-problems and so on until the problems are sufficiently simple that no further breaking down is needed. [2]

b) i) Syntax error,
Breaking of the rules of the language.
ie errors pertaining to the grammar of the programming language.
Example (using C++): not ending a statement with a semi-colon is a syntax error
i = i+ 1 // not ending a statement with a semi-colon [2]

ii) Logic error,
Error or flaws in the logic of the program or algorithm that causes the program to produce an incorrect output. Usually caused by not using control structures correctly
Eg, If x<=a is written as x<a, logic error has occurred [2]

iii) Arithmetic error,
It arises from errors in arithmetic computation. Can usually be spotting by looking at the arithmetic statements in the source codes, especially where the parenthesis are placed.
Eg, the statement a+b/c would result in a+(b/c) even though programmer may be thinking of (a+b)/c.
A potential arithmetic error in the code was 'b/c' if c=0. [2]

iv) Semantic error.
Semantics is the study of meaning of a programming language.
Semantic error is writing a valid program structure with invalid logic or inconsistency.
Example s=t+2
is a semantic error if s had been declared as a char and t is an integer. [2]

c)
- Presence of debugger which can pinpoint the exact line where the syntax error occurs
- Can trace /step through the program.
- Breakpoints can be placed at different points in the program
- Has watch windows to see status (values of variables, arrays etc) of program execution
- provide hints/suggestions to rectify error.

Any 3 [3]

d)
- Use of blank lines: Different sections of code can be identified clearly.
- Indentation: It is important as it will show the structure of the whole program clearly and thus improve readability.
- Comments and annotations: Help to explain to the reader what each section of codes does and it will be easier for reader to understand the algorithm.
- Meaningful variable names : ensure the purpose Is clear and thus enhances clarity of the algorithm

[any 2 with description] [4]

3 a) i) A true or false value, the computer stores 1(or any other values except 0 in C++) as true, and 0 as false.

Example in C++
bool isprime It is set to true if a number is a prime number, false otherwise. As it has only true or false answer, Boolean is appropriate.

ii) Integer data type contains a whole number.
It is used to store the values of the current number in stock. Integer is used instead of real because an item has to be a whole number.
eg int stockno
Or to use as a loop counter. For (int i=0; i<10; i++)

iii) Real data type contains a floating point value
Real: It is used to store the values for the price. As price has decimal value, real data type is appropriate.
Or amount of money, wages , weight , height etc.

iv) Character
Use to store a single character
Example gender : M or F [8]

b)
- integers need less memory,
- integer arithmetic runs faster,
- only the use of an integer as a loop counter can ensure correct termination. [2]

c) in C++ enumerated data types [3]

```
int main()
{
    enum days_of_week { Sun, Mon, Tue, Wed, Thu, Fri, Sat };
     days_of_week day1, day2;    // define variables of type days_of_week
     day1 = Mon;
     day2 = Tue
     if (day1 < day2 )
     cout << " day1 comes before day2 \n";
          :
}
```

d) i) Pass by value-- the actual value of a variable is passed into the function. The value would not change when exiting from the function. When parameters are passed by value, copies of the original parameters are created, which will be destroyed upon completion of the procedure. For example, a simple procedure called Area(r), takes radius of a circle, calculates and outputs its area, then passing by value is used so that the original value of r is expected to be unchanged. [4]

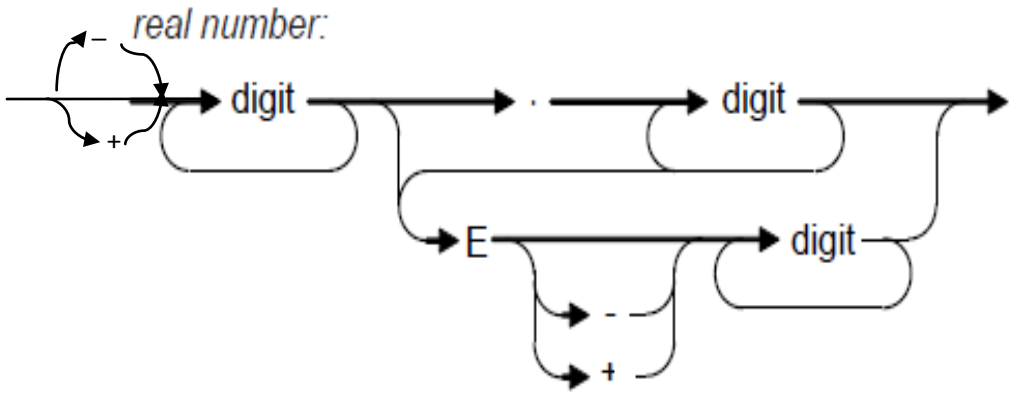
ii) Pass by reference (i.e. passing by address) -- The address of a variable is passed into the function. The value of the variable may be changed (usually change) after the execution of the function. When parameters are passed by reference, the addresses of the original parameters are sent to the procedure so that variables in the procedure share the same memory addresses as the passed in variables. This enables any changes made to the parameters during the execution of the procedure to be maintained after the procedure is completed. For example, a procedure called Swap(a, b), takes in two integers a and b and swaps them. Only when using passing by reference, the changes due to swapping can be maintained.

4 a) i) -12.87 [2]
Not valid. No Minus sign for real number

ii) 5.92 [2]
Not valid. Only single digit is allowed after decimal point

iii) 22E+12 [2]
Not valid. Only single digit is allowed after + sign.

b)



[3]

c) <real number> ::= <signunsign integer>.<integer> | [5]
<signunsign integer>E<signunsign integer>|
<signunsign integer>.<integer> E<signunsign integer>
<signunsign integer>::=<integer>.<integer> |<sign><integer>.<integer>
<integer> ::= <digit> |<digit><integer>
<sign> ::= + | -

5) a) A recursive routine is a routine that invokes (call) itself by dividing the problem into smaller instances of the same problem until they can be solved (anchor point/base case or terminal condition is reached) and the routine reassembled to produce a solution of the original problem. [2]

b) i) The return address.
stack [1]
-return address placed on stack
-along with values of parameters
-parameters read off stack by procedure
-any returning values placed on stack by procedure
-return to address at top of stack at end of procedure.

ii) Stack. When a procedure is call return address, return value and values of parameters are pushed in the stack in the Last In First Out Order (LIFO). On return the values are popped out with return address enabling the procedure to return to the calling program [3]

c)
```
fibo(int n)
{
    if (n == 1) || (n == 2)
      return 1;
    else
      return fibo(n – 1) + fibo(n – 2);
}
```
[4]

d) i)
```
//Initialise number and sum]
  n = 98
  sum = 0
 //Repeat loop list and decrement the numbers]
  Repeat
     Print n
      sum = sum + n
      n = n – 2
  Until n  < 1
  Output sum
```
[3]

d) ii)
```
//Initialise number and sum
 n = 98
 sum = 0
 While n >0
  {
            print n
            sum = sum + n
            n = n – 2
   }
  Output sum
```
[3]

Main difference is Repeat loop will be executed at least once, whereas the while loop may or may not be executed at all. [1]

6 a) i) A specialized language which allow users to ask for the information to be extracted from a database. Query language is a simplified programming language restricted to querying a database. Commonly used functions are selecting, searching, interrogating etc.
For example, the query

SELECT * FROM CUSTOMER WHERE AGE > 30

Request all records in which the age field has a value greater than 30.

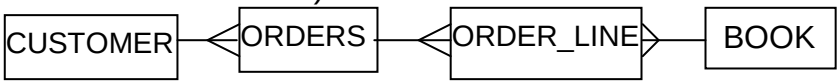
ii) Report generators are useful for the generation of reports They help the programmer to **design and produce the necessary reports** with easy-to-user interface by putting certain components on a report form. It allows users to define report definition, report format,report layout, display order etc.

iii) It allows database users to generate input form on screen for input new data to the database and the output facilities make possible the display the data and other information on the screen. An input/output screen generator is a program that allows the user to design input/output screens with standard window components such as text boxes, bullets, buttons, text labels, etc. This is usually done by dragging visual components and positional them on the interface. The code is then automatically generated.

b) i) Customer(<u>CustomerID</u>, Name, Address, ContactNo)
Order(<u>OrderNo</u>, CustomerID, DatedOrder, DeliveryType)
OrderLine(<u>OrderNo,ISBN</u>, Quantity)
Book(<u>ISBN</u>,Title, Author, Publisher Price, Quantity)
Note : Primary key is underlined, for OrderLine : the key is composite key consist of OrderNo and ISBN [8]

ii) The foreign key in one table will link to the primary key in another table with the following relationship [4]

- The CustomerID from CUSTOMER table can be linked to the Customer ID in the ORDERS table with a one to many relationship (one customer can have many orders)
- The OrderNo in the ORDERS table can be linked to the ORDER No in the ORDER_LINE table in a one-to-many relationship (Each order may have many lines on it, each for a different product.)
- The ISBN in the ORDER_LINE can be linked to the ISBN in Book table in many-to-one relationship (many orders can be for the same book)



iii) Problem:
1. Record updated by two users simultaneously (at the same time;)
2. First update is overwritten one update getting lost;

Solution
1. Time stamping;
Last update time of record on server must be earlier than read time of workstation attempting update;
2 Record locking;
record locking; so 2nd user has record available as read-only; when 1st user has it open in R/W mode;
only allow one user to edit at a time;

[3]

7 a) Steps:

1. Take first element as a pivot and partition the current table into 2 subtables so that those elements on the left are smaller than the pivot and those on the right larger than the pivot (see description below on how to place the pivot). [6]
2. Invoke quicksort to sort the left subtable.
3. Invoke quicksort to sort the right subtable.

Technique of placing the pivot (partitioning element) :
Use two index variables, i starting from the left of the sequence and j starting from the right. First i is incremented until it references an item greater than the partitioning element, and the j is decremented until it references an item less than the partitioning element. This pair of items is exchanged and the cycle is repeated. When i>=j the proper place to insert this partitioning element is obtained (ie swap portioning element with element at location j).

99, 87, 91, 96, 105, 112, 97, 81, 100, 93

1) Taking 99 as pivot, i increment (ie i=5 and j=10
   i=5, j=10 99, 87, 91, 96, <u>105</u>, 112, 97, 81, 100, <u>93</u>
   the elements at location i and j are swapped (see list below)
   K[i] <->K[j] 99, 87, 91, 96, ***93***, 112, 97, 81, 100,***105***
2) i increment until it references an element (112 at location 6) larger than pivot(99). J decrement until it references an element (81 at location 8) less than the pivot
   i=6, j=8 99, 87, 91, 96, 93, <u>112</u>, 97, <u>81</u>, 100, 105
   the elements 112 and 81 are then swapped (see list below)
   K[i] <->K[j] 99, 87, 91, 96, 93, ***81***, 97, ***112***, 100, 105
3) i increment until it references an element larger than the pivot (i= 8 ,element 112), j decrement (j=7, element 97)
   i=8, j=7 99, 87, 91, 96, 93, 81, <u>97</u>, <u>112</u>, 100, 105
   At this point since i>j, portioning element is swapped with K[j]
   Ie 99 and 97 swapped

   K[LB]<->K[j] **97**, 87, 91, 96, 93, 81, **99**, 112, 100, 105

4) 99 (the pivot) is now at proper location where left subtable is smaller and right subtable is larger

   {97, 87, 91,96, 93,81}, **99**, {112, 100, 105}
5) Repeat the procedure with left subset
6) Repeat the procedure with right subset

b) i) 81,87, 91, 93, 96, 97,99, 100, 105, 112,

low	high	mid	element
1	10	5	96
6	10	8	100
6	7	6	97
7	7	7	99

96, 100, 97, 99

ii)

low	high	mid	element
1	10	5	96
1	4	2	87
3	4	3	91
4	4	4	93

96, 87, 91, 93 [2]

c) 2^{11} (2048) < 3000 <2^{12} (4096)

12 times [1]

d)
```
Lower = 1
Upper = N
FOUND = false
Repeat
    Mid = Int ((Lower + Upper ) / 2 )
    If X[Mid] equals search_value Then
        FOUND = true
    Else
        If X[Mid] < search_value Then
            Lower = Mid + 1
        Else
            Upper = Mid – 1
        End if
    End if
Until ( (Lower > upper) or FOUND )
If FOUND Then
    Output “element in the array”
Else
    Output “ element not in the array”
```
[5]