1. a.) Define software. Explain its types.

It is a computer program which is a sequence of instructions designed to direct a computer to perform certain task. A software is an interface between user and computer. It is responsible for controlling, integrating and managing the hardware components of a computer system and for accomplishing specific task.

Software Types:

1.1 System Software

System software consists of several programs, which responsible for controlling, integrating and managing the individual hardware components of a computer system. System software is generally prepared by the computer manufacturers.

Examples: operating system

1.2 Application Software

Application software is specific purpose software which is used by user for performing specific task. It includes set of programs that do real work for users. There are two types of application software.

Types: general purpose software and specific purpose software.

1.b.) What are the steps required to develop a computer program? Explain.

The steps required to develop a computer are:

1. Problem Analysis.
2. Algorithm development.
3. Flowchart development
4. Coding.
5. Compilation and Execution.
6. Debugging and Testing.
7. Documentation.

1.1 Problem Analysis

At first you try to solve manually. If it is solvable manually by using your idea and knowledge, then you can use such idea and principle in programming and solve the problem by using computer.

Problem analysis should clearly specify the following tasks.

- Objectives
- Output requirement
- Input requirement
- Processing requirement
- Evaluating feasibility

1.2 Algorithm development and flowcharting

Algorithm

An algorithm is the step by step description of the procedure written in human understandable language for Solving given problems.

Flowchart

A flowchart is a pictorial representation of an algorithm that uses boxes of different shapes to denote different types of instructions. The actual instructions are written within these boxes using clear and concise statements.

1.3 Coding

code is a set of instruction that a computer can understand.

1.4 Compilation & Execution

The process by which source codes of a computer (programming) language are translated into machine codes is known as compilation. After compilation if everything is ok, the code is going under other process that is known as execution.

1.5 Debugging and Testing

It is the process of detecting and removing errors in a program, so that the program produces the desired results on all occasions. Before going in details about debugging and testing process, different errors that can occur should be discussed.

1.6 Testing

It is the process of executing a program or system with the intent of finding errors. It involves any activity aimed at evaluating an attribute or capability of a program or system and determining that it meets its required results.

2. a.) what are relational operators and assignment operators? Explain with examples.

Relational operator are the operators which are used to compare arithmetic, logical and character expression. A relational operator checks the relationship between two If the relation is true, it returns 1; if the relation is false, it returns value 0

Examples:

operator	meaning
<	is less than
<=	Is less or equal to
>	Is greater than
>=	Is greater or equal
==	Is equal to
!=	Not equal to

An assignment operator is used for assigning a value to a variable.

Examples =,+=,-=,*=

2. b.) Rewrite the following program by correcting any syntactical errors, if present. Also show the output of the

Corrected code.

```
#include<stdio.h>
#include<math.h>
int main()
{
   float root; int i=1,sum;
   do{
       sum=2*i-1;
      printf("\t%d\n",sum);
      i*=5/3;

   }while(sum<=15);
   root= pow(i,1/2);
   printf("\n%3f",root);
   return 0;


}
```

Output:

```
      -1

1.000000
```

3. a.) write a program to read the number until -1 is encountered. Also count the number of even number and odd numbers entered by the user.

```
#include<stdio.h>
void main(){
   int n,e=0,o=0;
```

```
   printf("enter the numbers");
   do{
      scanf("%d",&n);
      if(n%2==0)
      {
         e+=1;
      }
      else{
         o+=1;
      }
   }while(n!=-1);
   printf("\n the %d odd and %d even",o,e);
}
```

```
Output
enter the numbers1 2 3 4 5 -1

 the 4 odd and 2 even
```

3. b.) Distinguish between break and continue statement with example.

Break	Continue
Break is used to break loop or iteration.	Continue continues the loop or iteration.
Used with switch case loop.	Not used with switch case.
Keyword used is "break".	Keyword used is "continue".
Breaks loop and allows coming out from it.	Allows iterating in the loop.
Control is transferred outside the loop.	Control remains in the same loop.

E.g.
 	
for(i=1; i<=5 ; i++)
{
if(i==3)
break;
Printf("%d\n",i);
}
Output: 1, 2
	E.g.
	
	For(i=1 ;i<=5; i++)
	{
	if(i==3)
	continue;
	printf("%d\n",i);
	}
	Output: 1, 2, 4, 5

4.a.) Explain how the function are defined in c? Differentiate call by value and call by reference.

In C programming, functions are declared with a signature that includes the function name, return type, and parameter types. The function definition contains the implementation of the code. Functions promote code modularity and reusability, allowing you to call them from other parts of the program.

Syntax: return-type function_name(type1 argument1, type2 argument2, type3 argument3,……..typen argument n);

Call by value	Call by reference
In call by value, we pass the values of variables to it.	In call by reference, instead of passing the values of variables, we pass the address of variables.
does not allow to make any changes in the actual variables	Allows you to make changes in the values of variables using function calls.
Actual and formal arguments will be created in different memory location	Actual and formal arguments will be created in same memory location.

Values of variable are passed using normal variable	Pointer variables are required to store the address of variables.
Syntax of calling function: Function_name(variable_name1, variable_name2….);	Syntax of calling function Function_name(&variable_name1,&variable_name2,…..._;

4. b.) Write a program using a function that returns the largest number from an array of numbers that is passed to the

Function.

```
#include<stdio.h>
int large(int a[100],int n)
{
   int i,temp=0;
   for(i=0; i<n; i++)
{
if(a[i]>temp)
{
   temp=a[i];
}
}
return temp;
}
void main()
{
   int a[100],n,i=0;
   printf("enter the number of numbers you want to compare");
   scanf("%d",&n);
   printf("enter all the numbers seperated by space");
   for(i=0; i<n; i++)
   {
      scanf("%d",&a[i]);
   printf("the largest number is %d",large(a,n));


}
   }
```

Output:
enter the number of numbers you want to compare2
enter all the numbers seperated by space1 2
the largest number is 2

5. a.) How are the one dimensional and two dimensional arrays created in c? Explain with example.

Elements of one dimensional array can be represented either as a single column or single row. One dimensional a Is specified with a single subscript to locate a specific element.

Syntax: datatype variable_name[size of array]

Example: int a[10]; it is an 1D array consisting 10 memory location allocated for elements of integer datatype

Two dimensional array can be seen as an array of array. A pair of indices representing the position of the element in the array can be used to access each element of the array.

Syntax: data type array_name[row_size][column_size];

Example: int a[10][10] ;

It is an 2D array of integer datatype consisting 10 number of rows and 10 columns.

5. b.) write a c program to read two matrices from the user, add them and display the result in matrix form.

```
#include<stdio.h>
void main()
{
   int m,n;
   printf("enter the number of rows and columns\n");
   scanf("%d%d",&n,&m);
   int a[m][n],b[m][n],sum[m][n];
   for(int i=0; i<m; i++)
   {
      for(int j=0; j<n; j++)
      {
         printf("enter the %d%d element of a matrix",i+1,j+1);
         scanf("%d",&a[i][j]);
      }
   }
    for(int i=0; i<m; i++)
   {
      for(int j=0; j<n; j++)
      {
         printf("enter the %d%d element of b matrix\n",i+1,j+1);
         scanf("%d",&b[i][j]);
      }
   }
   printf("the sum is\n");
   for(int i=0; i<m; i++)
   {
      for(int j=0; j<n; j++)
      {
         sum[i][j]=a[i][j]+b[i][j];
         printf("%d\t",sum[i][j]);
```

```
    }
    printf("\n");
```

```
Output:
enter the number of rows and columns
2 2
eter the 11 element of a matrix1
enter the 12 element of a matrix1
enter the 21 element of a matrix1
enter the 22 element of a matrix1
enter the 11 element of b matrix
2
enter the 12 element of b matrix
2
enter the 21 element of b matrix
2
enter the 22 element of b matrix
2
the sum is
3    3
3    3
```

6. What do you mean by nested structures? Give suitable example. Write a program to read the heights of two Students and display the difference between their heights. Use feet and inches as members of structures to define height.

Nested structure means the structure within the structures.

Example:

```
struct employee
        {
        char address[20];
        float salary;
         struct person
        {
          }s;

           } p;
```

```
#include <stdio.h>


struct height {
  int feet;
  float inch;
} d1, d2, result;


int main() {
  printf("Enter height of first person\n");
  printf("Enter feet: ");
```

```
  scanf("%d", &d1.feet);
  printf("Enter inch: ");
  scanf("%f", &d1.inch);

  printf("\nEnter height of second perosn\n");
  printf("Enter feet: ");
  scanf("%d", &d2.feet);
  printf("Enter inch: ");
  scanf("%f", &d2.inch);
  result.feet = d1.feet - d2.feet;
  result.inch = d1.inch - d2.inch;

  // convert inches to feet if greater than 12
  while (result.inch >= 12.0) {
    result.inch = result.inch - 12.0;
    ++result.feet;
      }
  printf("\ndifference of height = %d\'-%.1f\"", result.feet, result.inch);
  return 0;
}
```

```
Output:
Enter height of first person
Enter feet: 11
Enter inch: 1

Enter height of second perosn
Enter feet: 11
Enter inch: 1

difference of height = 0'-0.0"
```

7. a.) Compare array and pointer with example.

An array is the collection of multiple items of the same type stored in contiguous memory locations. While declaring arrays, the size should be mentioned and their indexing starts from 0.

datatype var_name[size_of_array] = {elements};

example: int a[10];

A pointer is the symbolic representation of addresses. It stores the address of variables or memory location. Pointers enable programmers to create and manipulate dynamic data structures. Variables can be of type int, array, char, function, or any other pointer.

Syntax: datatype *var_name;
Example: int *a
b=10;
a=&b;

7. b.) Write a program to read a string from user and use a user defined function to copy the content of read string into another character array changing lower case letter to upper if any. Use pointer to process the string.

```
#include <stdio.h>


void copy(char *source, char *destination);


int main()
{
   char *string1[100];
   char *string2[100];


   printf("string\n");
   scanf("%s", string1);


   printf("original %s\n", string1);
   copy(string1, string2);
   printf("copied  %s\n", string2);



   return 0;
}


void copy(char *source, char *destination)
{
   while (*source != '\0')
   {
*destination=*source-32;
      source++;
```

```
    destination++;
  }
  *destination = '\0';
}
```

Output:

```
string
sosyika
original sosyika
copied  SOSYIKA
```

8. Write a program to read the details of book authors and write it to a file, until user confirms to end. Then read and display the nth record in the file, where n is read from the user. The data for authors must be represented by structures that contains name, nationality and number of books published.

```
#include<stdio.h>

void main()
{

  struct author{

  char name[20];
  char nat[25];
  int bok;
  int number;

  }book;
  FILE *f1 ;
  char cont;
  f1=fopen("books.txt","w+");
  int i=0;
  do {
    printf("enter the record for book no %d\n",i+1);
    printf("enter the name\n");
    scanf("%s",&book.name);
```

```
        printf("enter nationality:\n");
        scanf("%s",&book.nat);
        printf("enter the publishid book no:");
        scanf("%d",&book.number);
fprintf(f1,"%d. Name=%s nationl=%s number=%d \n",i+1,book.name,book.nat,book.number);
printf("do you want to add more:\n");
scanf(" %c",&cont);
i++;
}while(cont=='y' || cont=='Y' );
fclose(f1);
   f1=fopen("books.txt","r");
   int bok=i;
   int k=1;
      printf("enter the record no you want to search\n");
      scanf("%d",&bok);

     while( fscanf(f1,"%d. Name=%s nationl=%s number=%d \n",&k,&book.name,&book.nat,&book.number)==4){
      if(k==bok){
         printf("%d. Name=%s nationl=%s number of book published=%d ",bok,book.name,book.nat,book.number);
         break;

      }
 }
}
```

```
Output:
enter the record for book no 1
enter the name
john
enter nationality:
british
enter the publishid book no:5
do you want to add more:
y
enter the record for book no 2
enter the name
hoover
enter nationality:
```

```
american
enter the publishid book no:4
do you want to add more:
n
enter the record no you want to search
1
1. Name=john nationality=british
number of book published=5
```

9.a) Explain the FORTRAN structure. What are the data types in FORTRAN.

FORTRAN Structure: the main parts of fortran are:

- Program name
- Declaration of variable
- Initial definition of variable
- Main program body
- End

Example: integer a,b,c
C=a+b
Write(*,*)'c=',c
end

Data Types in FORTRAN: FORTRAN supports integer, real (floating-point), character (text), logical (boolean), complex (numbers with real and imaginary parts), and derived types.

9.b) write a program in FORTRAN to solve quadratic equation and display roots in proper format.

PROGRAM QuadraticSolver

```
  REAL :: a, b, c, discriminant, root1, root2

  ! Input coefficients a, b, and c
  WRITE(*, *) "Enter coefficient a:"
  READ(*, *) a
  WRITE(*, *) "Enter coefficient b:"
  READ(*, *) b
  WRITE(*, *) "Enter coefficient c:"
  READ(*, *) c

  ! Calculate the discriminant
  discriminant = b**2 - 4*a*c

  ! Check the nature of the roots
  IF (discriminant > 0) THEN
    root1 = (-b + SQRT(discriminant)) / (2*a)
    root2 = (-b - SQRT(discriminant)) / (2*a)
    WRITE(*, *) "Root 1:", root1
    WRITE(*, *) "Root 2:", root2
  ELSEIF (discriminant == 0) THEN
    root1 = -b / (2*a)
    WRITE(*, *) "Repeated root:", root1
  ELSE
    WRITE(*, *) "Complex roots."
  END IF
```

END

Submitted by:
079bei026: PRAZWAL CHAPAGAIN
079bei010: AUSTINA ARYAL
079bei041: SOSTIKA SHRESTHA