Question 1

a) In physics, an object that is in motion is said to have kinetic energy. The following formula can be used to determine the kinetic energy that a moving object has:

KE = ½ mv^2

The variables in the formula are as follows:

KE – kinetic energy in joules
m – object's mass in kilograms
v – object's velocity in meters per second.

Write a function named kineticEnergy that accepts an object's mass (in kilograms) and velocity (in meters per second) as arguments. The function should return the amount of kinetic energy that the object has.

In main(), demonstrate the function by calling it (by value) in a program that asks the user to enter values for mass and velocity. Display the output.

Sample Output Screen
Enter an object's mass and velocity as required.... Mass in kilograms: *45.67* Velocity in meters per second: *12.2* The kinetic energy of this object is 3398.76 joules.

b) Referring to question a), the function is called by value. Modify the solution so that the function to be called by reference using pointers. The new function prototype is given as:

```
void kineticEnergy(double*, double*, double*);
```

The function will be passed 3 variables by reference which are kinetic energy, object's mass (in kilograms) and velocity (in meters per second).

c) Refer to question b). Modify the solution so that the function to be called by reference using reference arguments. The new function prototype is given as:

```
void kineticEnergy(double&, double&, double&);
```

The function will be passed 3 variables by reference which are kinetic energy, object's mass (in kilograms) and velocity (in meters per second).

Question 2

The following function will use a, b, and c as the coefficients of a quadratic equation to compute b^2 - 4ac (also known as discriminant). This function calls on another function called get_a_b_c to get the values for a, b, and c from user input. [Note: function call by reference using reference arguments since the variables in the function to holds the value is needed by the other function]

Write the complete program, compile and run it.

```
double bb_4ac( )
{
        double a, b, c;  // Coefficients of a quadratic equation
        get_a_b_c(a, b, c);
        return b*b - 4*a*c;
}
```

Sample Output Screen

Enter a, b and c: *1 3 1*
The discriminant is 5

Note: The inputs are separated by a space, eg: 1[space]3[space]1[space][enter]

Question 3

a) You are required to write a program that calculates the grade of students. Each student will have **5** subjects. You are required to get the student's *name* and *marks* from the user. The marks must be kept in a one dimensional array in the main() function. This array must then be passed to an **average(….)** function in order to calculate the average marks for students. The average mark that has been calculated must then be returned to the main() function.

AVERAGE MARKS	*GRADE*
average_marks above= 80	A
average_marks above =60	B
average_marks above =50	C
average_marks below 50	F

Next, from the main() function, the average is passed to the **greds(…)** function to calculated the grade for students. Grade that has been calculated must be returned to the main() function. In the main() function, display the student's name, average mark and grade. Output should be as follows.

Sample Output Screen

```
Enter Name :Dory
Enter Marks : 78
Enter Marks : 55
Enter Marks : 50
Enter Marks : 91
Enter Marks : 55

Name    : Dory
Average : 65.8
Grade   : B
```

b) Based on 3(a), modify the answer to include a record called *Student* as given below:

```
struct Student
{
      char name[30], grade;
      float marks[5], avg;
};
```

Therefore, in main(), all the variables declared (except the counter variable), will be grouped under the record. Declare a variable *S1* of the record *Student* type and modify the other statements in your main(). Note that the functions you've included in your answer at 3(a) will not have any changes.

Question 4

Write a complete program based on the following requirements:

- Use the following record:

```
struct Rental
{       char name[20];
        float rent;
        int month;
};
```

- Declare and define function *report(..)*:
  - Parameter : array *r* of record *Rental* [size 3]
  - Function prototype given : float report(struct Rental[]) ;
  - Using a for loop, display on the screen the records of tenants that have rental due more than RM1000. Also calculate the total amount due for these tenants. **[Note: refer to sample output screen]**
  - Return the total amount due.

- In ***main()***,
  - Declare an array *r* of record *Rental.* Array size is 3. You may use the code below that has hardcoded values for it.

```
struct Rental R[3] = {{"Megan", 300, 4},{"Johnson", 250, 1},
                      {"David", 790, 2}};
```

  - Call function *report(...)*:
  - Parameter : array *R* of struct *Rental.*
  - The function will return the grand total.
  - Display the grand total amount that is due.

Sample Output Screen

```
--------------------------------------------------
-          AMOUNT DUE MORE THAN RM1000.00        -
--------------------------------------------------
Tenant name    : Megan
Monthly rental : RM 300.00
Unpaid months  : 4
Unpaid amount  : RM 1200.00

Tenant name    : David
Monthly rental : RM 790.00
Unpaid months  : 2
Unpaid amount  : RM 1580.00

Total rental to be collected : RM 2780.00
```