

```
/* program to calculate the solution to a quadratic equation.
   Only non-complex solutions are found */

#include <stdio.h>
#include <math.h>

// function prototype
int findx(double, double, double, double *, double *);

int main(void)
{
    double a, b, c;           // coefficients
    double highx, lowx; // two values of x
    int result;              // returned value from findx

// introduction to program (cout could be used here)
    printf("Enter coefficients of quadratic equation");
    printf(" of form ax^2+bx+c=0\n");

// the user enters the three coefficients (no error trapping)
    printf("Enter a : ");
    scanf("%lf",&a);
    printf("      b : ");
    scanf("%lf",&b);
    printf("      c : ");
    scanf("%lf",&c);

// function which calculates the solution (two values)
    result=findx(a, b, c, &highx, &lowx);

// output the solution to screen (depends on solution)
    if(result==0) {
        printf("x is either %f or %f\n\n", highx, lowx);
    } else if (result==1){
        printf("x is %f\n\n", highx);
    } else {
        printf("Equation has complex solution\n\n");
    }
}
```

Setter: .....AJS

Assessor:.....

Coordinator:.....AJS

UNIVERSITY OF LEICESTER ENGINEERING DEPARTMENT

Second Year September Examinations 2001

Module No. EG222

Module title: PROCESS ORIENTED PROGRAMMING

Solution No:

Page No.

Date: 23 July 2001

--

Setter: .....AJS

Assessor:.....

Coordinator:.....AJS

```
int findx(double a, double b, double c, double *highx, double *lowx)
{
    int result;
    double temp;

    if(a==0) { // check if a is zero, in which case x=-c/b
        *highx=-c/b;
        *lowx=0;
        result=1; // in specification
    } else { // otherwise use usual equation (see data book)
        temp=b*b-4.0*a*c;
        if(temp>=0){ // check whether argument to sqrt is positive
            *highx=(-b+sqrt(temp))/(2*a);
            *lowx=(-b-sqrt(temp))/(2*a);
            result=0;
        } else { // square root of a negative number (complex solution)
            *highx=0.0;
            *lowx=0.0;
            result=2;
        }
    }
    return result;
}
```

Setter: .....AJS

Assessor:.....

Coordinator:.....AJS

UNIVERSITY OF LEICESTER ENGINEERING DEPARTMENT

Second Year September Examinations 2001

Module No. EG222

Module title: PROCESS ORIENTED PROGRAMMING

Solution No:

Page No.

Date: 23 July 2001

--

Setter: .....AJS

Assessor:.....

Coordinator:.....AJS

```
#include <stdio.h>

// function prototypes
int read_data(int *, float *);
float find_mean(int, float *);
float find_low(int, float *);

int main(void)
{
    int number_data, fileok;
    float data_array[1000], mean, lowest_value;

// read the data file
    fileok=read_data(&number_data, data_array);

    if(fileok==0){        // data have been read OK
        mean=find_mean(number_data, data_array);
        lowest_value=find_low(number_data, data_array);
        printf("Mean = %f\n", mean);
        printf("Lowest value = %f\n", lowest_value);
    }
}

int read_data(int *number_data, float *data_array)
{
// read data in from file, c++ commands could be used instead

    FILE *dataptr;
    int fileok, ii;

    dataptr=fopen("numbers.dat", "r");    // open file for reading

    if(dataptr==NULL) { // check that file has opened
        printf("Error in opening file\n");
        fileok=1;
    } else {
        fscanf(dataptr, "%d", number_data);
        if(*number_data>=1000){ // not a DMA question, array size is fixed
            printf("Too many data points in file\n");
            fileok=2;
        } else { // read in data from file into array
            for(ii=0; ii<*number_data; ii++){
                fscanf(dataptr, "%f", data_array+ii);
            }
            fileok=0;
        }
        fclose(dataptr); // must remember to close file
    }

    return(fileok);
}
```

Setter: .....AJS

Assessor:.....

Coordinator:.....AJS

UNIVERSITY OF LEICESTER ENGINEERING DEPARTMENT

Second Year September Examinations 2001

Module No. EG222

Module title: PROCESS ORIENTED PROGRAMMING

Solution No:

Page No.

Date: 23 July 2001

Setter: .....AJS

Assessor:.....

Coordinator:.....AJS

```
float find_mean(int number_data, float *data_array)
{
    float running_total, mean;
    int ii;

    for(ii=0; ii<number_data; ii++){
        running_total=running_total+data_array[ii];
    }
    mean=running_total/number_data;

    return(mean);
}

float find_low(int number_data, float *data_array)
{
    float lowest_value;
    int ii;

    // assume that data is never higher than this value
    lowest_value=1.0e6;

    for(ii=0; ii<number_data; ii++){
        if(data_array[ii]<lowest_value){
            lowest_value=data_array[ii];
        }
    }

    return(lowest_value);
}
```

Setter: .....AJS

Assessor:.....

Coordinator:.....AJS

UNIVERSITY OF LEICESTER ENGINEERING DEPARTMENT

Second Year September Examinations 2001

Module No. EG222

Module title: PROCESS ORIENTED PROGRAMMING

Solution No:

Page No.

Date: 23 July 2001

Setter: .....AJS

Assessor:.....

Coordinator:.....AJS

(a) The essential points are:

- (i) Pointers : hold an address, used in call-by-address
- (ii) Call by address : A method of ensuring that changes to a variable made in a function are returned to the calling function. Useful when more than one variable needs to be returned.
- (iii) Dynamic memory allocation: Allows the storage (ie size) of arrays to be determined at runtime rather than at compile time. Useful when the number of elements in an array is not known at runtime and efficient use of memory is required.
- (iv) Structs: Allows the definition of new data types, particularly useful for combining related data (e.g. all the information about a person - name, age, etc.)

(b) The corrected program is shown below (points of correction are shown by [x])

```
// program to sort into alphabetical order some surnames      [0]

#include <stdio.h>
#include <string.h>

#define NUMNAMES 5      [1]
#define NUMCHAR 30     [2]

int main(void) {

    int ii, jj, kk;     [3]
    char test_names[NUMNAMES][NUMCHAR]={"Smith A.", "Jones P.", "Jackson Q.",
        "Smith C.", "Hamilton M."};
    char temp_name[NUMCHAR]; [4]

    for(ii=0; ii<NUMNAMES-1; ii++){ [5]
        for(jj=ii+1; jj<NUMNAMES; jj++){
            for(kk=0; kk<NUMCHAR; kk++){
                if(test_names[ii][kk]>test_names[jj][kk]){
                    strcpy(temp_name, test_names[ii]);
                    strcpy(test_names[ii], test_names[jj]);
                    strcpy(test_names[jj], temp_name); [6]
                    break; // from inner loop
                } else if (test_names[ii][kk]<test_names[jj][kk]) {
                    break; // from inner loop
                }
            }
        }
    }

    printf("The names in alphabetical order are :\n\n");
    for(ii=0; ii<NUMNAMES; ii++){
        printf("%s\n", test_names[ii]);
    }
}
```

Setter: .....AJS

Assessor:.....

Coordinator:.....AJS

UNIVERSITY OF LEICESTER ENGINEERING DEPARTMENT

Second Year September Examinations 2001

Module No. EG222

Module title: PROCESS ORIENTED PROGRAMMING

Solution No:

Page No.

Date: 23 July 2001

```
}  
}  
[7]
```

Setter: .....AJS

Assessor:.....

Coordinator:.....AJS

[0] Needs a few comments throughout

[1] No semi-colon required after #define

[2] Style and maintainability would be improved by #defining the number of characters the names are to contain.

[3] kk has not be declared

[4] Missing semi-colon

[5] Should be NUMNAMES-1 (otherwise the value of jj will reference an element beyond the end of the array)

[6] kk should be replaced by jj (referencing beyond the end of the array)

[7] Missing brace at end of main

Setter: .....AJS

Assessor:.....

Coordinator:.....AJS

UNIVERSITY OF LEICESTER ENGINEERING DEPARTMENT

Second Year September Examinations 2001

Module No. EG222

Module title: PROCESS ORIENTED PROGRAMMING

Solution No:

Page No.

Date: 23 July 2001

Setter: .....AJS

Assessor:.....

Coordinator:.....AJS