

**UNIVERSITY COLLEGE DUBLIN**

NATIONAL UNIVERSITY OF IRELAND, DUBLIN

An Colaiste Ollscoile Baile Atha Cliath

Ollscoil na hEireann, Baile Atha Cliath

---

**SUMMER EXAMINATIONS 2005**

---

**FIRST EXAMINATION IN ENGINEERING**

Programme Codes: ENBDF0002, ENBDF0003,  
ENBDF0004, ENBDF0005, ENBDF0008, ENBDF0011

**COMP1604 COMPUTER SCIENCE**

Prof. J. Kramer

Prof. B. Smyth

Dr. J. Murphy\*

**Time allowed: 2 hours**

Answer **Question 1** and **one** other Question.

Question 1 carries 60 marks; Questions 2 and 3 carry 40 marks.

This is a closed-book examination. No calculators allowed.

**Loose Rough Work sheets are not to be distributed or used.**

**READ EACH QUESTION CAREFULLY.**

### **Question 1 (COMPULSORY) [60 marks]**

Answer all parts (a) – (t). Each part carries 3 marks.

- (a) True or False (*no explanation required*): if a program compiles correctly then there will be no **run-time** errors in it.
- (b) True or False (*no explanation required*): in C, the programmer can not decide which **memory location** is to be used to store a particular variable in their program.
- (c) True or False (*no explanation required*): “**Three\_float**” is a valid identifier in C.
- (d) True or False (*no explanation required*): in C, the programmer can decide where the program will start execution, the default being **main ()**
- (e) What is the screen output of the following fragment of C code (*no explanation required*):

```
int i=5,j=-5;
if (j<1){
    printf("first\n");
} else if ((-i)<=(-j)){
    printf("second\n");
} else {
    printf("no match\n");
}
```

- (f) What is the screen output of the following fragment of C code (*no explanation required*):

```
float x=2.234907;
printf("value is %.3f\n",x);
```

- (g) What is the screen output of the following fragment of C code (*no explanation required*):

```
int i=90;
while (i>0){
    i = (i+1)/7;
    printf("i is %d\n",i);
}
```

- (h) What is the screen output of the following fragment of C code (*no explanation required*):

```
int a=0, b=2, c=-2;
if (a || ((b+c)>=0) ){
    printf("condition is true\n");
} else {
    printf("condition is false\n");
}
```

- (i) True or False (*no explanation required*): if a C function returns more than a single value from a **return** statement, this is detected by the compiler as a syntax error.
- (j) What is the screen output of the following fragment of C code (*no explanation required*):

```
double x[3] = {1.5, 2.2, 4.3};
printf("value is %.2f\n", *(x+1) );
```

Question 1 (continued)

- (k) What is the screen output of the following fragment of C code (*no explanation required*):

```
int j = -5;
do {
    j++;
    printf("j is %d\n", j);
} while (j < 0);
```

- (l) What is the screen output of the following C program (*no explanation required*):

```
#include <stdio.h>
int f1(int i, int j){
    return (-i*j);
}
int f2(int i){
    int j = f1(1-i, i-1);
    return j;
}
void main(void){
    printf("result is %d\n", f1(-1,f1(f2(f2(5)),3)) );
}
```

- (m) Select the **incorrect** answer: If you want to multiply the first and third elements of an array `arr[]` together, you should use

- (m-1) `arr[0]*arr[2]`
- (m-2) `(*arr)*(*(arr+2))`
- (m-3) `*arr[0]**arr[2]`

- (n) True or False (*no explanation required*): If `i` and `j` are integer variables and `arrflts[]` is an array of `floats`, `arrflts[i*j]` is a valid expression in C.

- (o) Select the correct answer: The function prototype

```
void fn(float *x);
```

tells us that

- (o-1) `fn()` takes no arguments and returns a value of type `float`.
- (o-2) `fn()` takes 1 argument of type `float` and does not return a value.
- (o-3) `fn()` takes 1 argument of type pointer-to-`float` and does not return a value.

- (p) Select the correct answer: `fopen("datafile.txt","r")` means

- (p-1) if it exists, open `datafile.txt` for random access, otherwise return an error.
- (p-2) if it exists, open `datafile.txt` for reading only, otherwise return an error.
- (p-3) if it exists, open `datafile.txt` for reading only, otherwise open the first file found and read from it.

Question 1 (continued)

- (q) What is the screen output of the following fragment of C code (*no explanation required*):

```
char str[]="abcdefghijklmn";
char vowels[]="aeiou";
int i, j;
for (j=0; vowels[j]!='\0'; j++){
    for (i=0; str[i]!='\0'; i++){
        if (str[i]==vowels[j]){
            str[i]='Y';
            break;
        }
    }
}
printf("string=%s\n", str);
```

- (r) True or False (*no explanation required*): to store the string "University", we need an array of type **char** with a size of at least 10 characters.
- (s) What is the screen output of the following fragment of C code (*no explanation required*):

```
double x[3];
int i, y[3];
for (i=0; i<3; i++){
    x[i] = 2.0/(i+1);
    y[i] = x[i];
}
printf("x[0] is %.2f\n", x[0]);
printf("x[1] is %.2f\n", x[1]);
printf("x[2] is %.2f\n", x[2]);
printf("y[0] is %i\n", y[0]);
printf("y[1] is %i\n", y[1]);
printf("y[2] is %i\n", y[2]);
```

- (t) Given the following definition and declaration:

```
struct Employee {
    int number;
    char name[30];
    int age;
    char position[30];
};
struct Employee emp1, emp2;
```

Which of the following statements correctly assigns the value 30 to **emp1**'s age?

- (t-1) `emp1.position = 30;`  
(t-2) `emp1->age = 30;`  
(t-3) `emp1.age = 30;`

## **Question 2 [40 marks]**

Answer all parts (a) – (c).

(a) The following fragment of C code outputs the digits in a string called **string** to the screen using a **for** loop:

```
int i;
char string[500];
/* assume string[] is somehow filled with letters, digits, etc */
for (i=0; string[i]!='\0'; i++){
    if ((string[i] >= '0') && (string[i] <= '9')){
        printf("%c", string[i]);
    }
}
```

*Re-write* this code fragment using a **while** loop instead of the **for** loop.

(b) The following fragment of C code uses **if** statements to determine whether to Sell, Hold, or Buy a stock based on its price:

```
if (stock_price >= 50){
    printf("Sell\n");
}
if ((stock_price >= 25) && (stock_price < 50)){
    printf("Hold\n");
}
if (stock_price < 25){
    printf("Buy\n");
}
```

*Re-write* this code fragment using **if/else-if/else** statements instead of the **if** statements.

(c) Consider the following C program:

```
#include <stdio.h>
/* DEFINITION OF FUNCTION "num_pos" GOES HERE */
void main(void) {
    int i, j=0, array1[8]={1,-1,-1,0,0,1,0,-1};
    j = num_pos(array1,8); /* function call to num_pos() */
    printf("there are %d positive elements of array1[]", j);
}
```

Write down the definition of function **num\_pos()** which counts the number of positive elements in its input array, so that the output of the above program is:

**there are 2 positive elements of array1[]**

### **Question 3 [40 marks]**

Answer parts (a) and (b).

- (a) Consider the following C program:

```
#include <stdio.h>
void main(void)
{
    int intarr[3], i;
    for (i=0 ; i<3; i++){
        printf("enter value number %d: ", i+1);
        scanf("%d", &intarr[i]);          /* LINE 1 */
        printf("you entered %d\n", intarr[i]); /* LINE 2 */
    }
}
```

*Re-write* the lines **LINE 1** and **LINE 2** using “array pointers” instead of array subscripts.

- (b) Consider the following C program:

```
#include <stdio.h>
void main(void)
{
    FILE *fptr1, *fptr2;
    int inp;
    fptr1 = fopen("input.dat", "r");
    fptr2 = fopen("output.dat", "w");
    while (fscanf(fptr1, "%d", &inp)==1){
        if (inp >= 0){
            fprintf(fptr2, "%d ", inp);
        }
    }
    fclose(fptr1);
    fclose(fptr2);
}
```

- (i) Suppose the file **input.dat** contains the following data:

```
0
-2
2
6
-1
-5
```

*Complete this sentence:*

After executing this program, the file **output.dat** contains \_\_\_\_\_ .

- (ii) *Re-write* the above program so that it writes all values between **-1** and **4** (inclusive) in the file **input.dat** to a new file called **output2.dat**

**o O o**