

**UNIVERSITY COLLEGE DUBLIN**

NATIONAL UNIVERSITY OF IRELAND, DUBLIN

An Colaiste Ollscoile Baile Atha Cliath

Ollscoil na hEireann, Baile Atha Cliath

---

**SUMMER EXAMINATIONS 2003**

---

**FIRST EXAMINATION IN ENGINEERING**

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

**COMP1604 COMPUTER SCIENCE**

Prof. J. Hughes

Mr. G. O'Hare

Dr. L. 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.

**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*): C is an example of a high-level programming language.
- (b) True or False (*no explanation required*): if a program compiles with no errors, it will run correctly when executed.
- (c) Complete the sentence: The two most widely used tools for developing algorithms are flowcharts and \_\_\_\_\_ .
- (d) If  $x==3$ ,  $y==3$ , and  $z==2$ , what is the value of  $w$  in the expression  $w = x - y * z$  ?
- (e) What is the screen output of the following fragment of C code (*no explanation required*):

```
float x=1.7320508;
printf("value is %.4f\n",x);
```

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

```
int i=-1,j=1;
if (j<=0){
    printf("normal case\n");
} else if ((-i)>=j){
    printf("exceptional case\n");
} else {
    printf("error case\n");
}
```

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

```
int a=0, b=2, c=-1;
if (a || ((b+c)>0) ){
    printf("condition is true\n");
} else {
    printf("condition is false\n");
}
```

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

```
int i=51;
while (i>0){
    i = i/4;
    printf("i is %d\n",i);
}
```

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

```
int i;
for (i=0;i<=4;i++){
    switch(i){
        case 1: break;
        case 2: printf("2\n");
                break;
        case 3: printf("3\n");
                break;
        default: printf("default\n");
                break;
    }
}
```

**[Question 1 continues]**

### Question 1 (continued)

- (j) Select the correct answer: If a function has a return type **void**, this means

(j-1) it is never executed.  
(j-2) it does not return a value.  
(j-3) it accepts no parameters.

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

```
int fname(double *x);
```

tells us that

(k-1) **fname()** takes 1 argument of type int and returns a value of type pointer-to-double.  
(k-2) **fname()** takes 1 argument of type double and returns a value of type int.  
(k-3) **fname()** takes 1 argument of type pointer-to-double and returns a value of type int.

- (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(i+1, i-1);
    return j;
}
void main(void){
    printf("result is %d\n", f1(-1,f1(4,f2(f2(4)))) );
}
```

- (m) Select the correct answer: If you want to add the first and third element of an array **arr**, you should use

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

- (n) True or False (*no explanation required*): If the value of a subscript is greater than the largest valid subscript of an array, it will cause a compiler error.

- (o) 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]);
```

*[Question 1 continues]*

### Question 1 (continued)

(p) Select the correct answer: If the pointer `ptr1` currently points to `x` and you want to assign the current value of `y` to `x`, you could use the statement

```
(p-1) ptr1 = &y;  
(p-2) *ptr1 = y;  
(p-3) y = *ptr1;
```

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

```
int x = 1;  
int y = -2;  
int* p = &x;  
*p = y*y + x*x;  
y = *p;  
printf("x is %d and y is %d\n", x, y);
```

(r) True or False (*no explanation required*): to store the string "program", we need an array of type `char` with a size of at least 8 characters.

(s) 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]){  
            vowels[j]='Y';  
            break;  
        }  
    }  
}  
printf("vowels=%s\n", vowels);
```

(t) Select the correct answer: `fopen("datafile.txt", "a")` means

- (t-1) open `datafile.txt` for appending, creating the file if it doesn't already exist.
- (t-2) open `datafile.txt` for appending if it exists, otherwise return an error.
- (t-3) open `datafile.txt` for all input/output operations.

## **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[100];
/* 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 converts a mark (an integer between 0 and 10 inclusive) into a grade using **if/else-if/else** statements:

```
if (mark>=7){
    printf("Excellent\n");
} else if (mark>=4){
    printf("Satisfactory\n");
} else{
    printf("Fail\n");
}
```

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

- (c) Consider the following C program:

```
#include <stdio.h>
/* DEFINITION OF FUNCTION "nonneg" GOES HERE */
void main(void) {
    int i, array1[8]={7,-8,-7,0,1,0,-1,1};
    nonneg(array1,8); /* function call to nonneg() */
    printf("\nelements of array1[] are ");
    for (i=0;i<8;i++){printf("%d ", array1[i]);}
}
```

Write down the definition of function **nonneg()** which replaces every negative element of its input array with a 0 value, so that the output of the above program is:

elements of array1[] are 7 0 0 0 1 0 0 1

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

Answer parts (a) and (b).

- (a) Consider the following C program:

```
#include <stdio.h>
void main(void)
{
    int intarr[4], i, total=0;
    for (i=0 ; i<4; i++){
        printf("enter value number %d: ", i+1);
        scanf("%d", &intarr[i]);          /* LINE 1 */
        total += intarr[i];              /* LINE 2 */
    }
    printf("sum of inputs is %d, first input was %d, last input
was %d\n", total, intarr[0], intarr[3]); /* LINE 3 */
}
```

*Re-write* the lines **LINE 1**, **LINE 2**, and **LINE 3** 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 >= 1 && inp <= 5){
            fprintf(fptr2, "%d,", inp);
        }
    }
    fclose(fptr1);
    fclose(fptr2);
}
```

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

```
0
2
-4
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 the **even** values in a file called **input2.dat** to a file called **output2.dat**

**o O o**