

UNIVERSITY COLLEGE DUBLIN

NATIONAL UNIVERSITY OF IRELAND, DUBLIN

An Colaiste Ollscoile Baile Atha Cliath
Ollscoil na hEireann, Baile Atha Cliath

AUTUMN EXAMINATIONS 2004

FIRST EXAMINATION IN ENGINEERING
Programme Codes: ENBDF0002, ENBDF0003,
ENBDF0004, ENBDF0005, ENBDF0008, ENBDF0011

COMP1604 COMPUTER SCIENCE

Prof. J. Kramer
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.

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*): a C program **must** be compiled into machine language to produce object code.
- (b) True or False (*no explanation required*): in C, program execution begins with the first statement of `main()`
- (c) True or False (*no explanation required*): if a program compiles with no errors, it will run correctly when executed.
- (d) True or False (*no explanation required*): `2_num` is a valid identifier in C.
- (e) What is the screen output of the following fragment of C code (*no explanation required*):

```
int i=5,j=-5;
if (j>-10){
    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*):

```
int a=0, b=2, c=-1;
if ((a-c) && ((b/3)>0) ){
    printf("condition is true\n");
} else {
    printf("condition is false\n");
}
```

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

```
int i=20;
while (i>0){
    i = (i-1)/2;
    printf("i is %d\n",i);
}
```

- (h) State the 3 types of errors that could be found in a C program (*no explanation required*).
- (i) Select the correct answer: If a function has a return type **float**, this means

- (i-1) it can manipulate floating-point variables only.
(i-2) it returns a floating-point value to the calling function.
(i-3) it takes a single parameter of type **float**.

[Question 1 continues]

Question 1 (continued)

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

```
void fn(int x);
```

tells us that

- (j-1) `fn()` takes 1 argument of type `int` and does not return a value.
(j-2) `fn()` takes no arguments and returns a value of type `int`.
(j-3) `fn()` takes 2 arguments and does not return a value.

- (k) 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-2);
    return j;
}
void main(void){
    printf("result is %d\n", f1(-1,f1(f2(f2(3)),2)) );
}
```

- (l) Select the correct answer: If you want to multiply the first and second elements of an array `arr[]` together, you should use

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

- (m) 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.

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

```
int a=3, b=-4, *ptr=&a;
b = *ptr;
printf("a is %d, b is %d\n", a, b);
```

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

```
int x = 0;
int y = 2;
int* p = &y;
*p = y+y;
x = (*p)*(*p);
p = &x;
printf("x is %d and y is %d\n", x, y);
```

[Question 1 continues]

Question 1 (continued)

(p) 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 %.1f\n", *(x+2) );
```

(q) True or False (*no explanation required*): to store the string "exam", we need an array of type **char** with a size of at least 5 characters.

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

```
int i;
char str[]="I hope to do well on this Exam.";
for (i=0; str[i+1]!='\0'; i++){
    if (str[i]==' ') {
        str[i] = '-';
    }
}
printf("%s\n", str);
```

(s) True or False (*no explanation required*): in a data file, the **only** way to tell how much data is in the file is to use a Sentinel signal (a value outside the range of actual data values) to indicate that the end of the data has been reached.

(t) Given the following definition and declaration:

```
struct Employee {
    int id_number;
    char name[30];
    int age;
    char position[30];
};
struct Employee emp1;
```

Which of the following statements correctly assigns the value 2123 to **emp1**'s **id_number**?

- (t-1) `emp1.id_number = 2123;`
- (t-2) `emp1->id_number = 2123;`
- (t-3) `emp1-id_number = 2123;`

Question 2 [40 marks]

Answer all parts (a) – (c).

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

```
int i;
char string[100];
/* assume string[] is somehow filled with letters, digits, etc */
i=0;
do {
    if (string[i] == '0'){
        printf("%c", string[i]);
    }
    i++;
} while (string[i]!='\0');
```

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

(b) The following fragment of C code classifies the voltage **v** using **if/else-if/else** statements:

```
int v;
/* suppose a value is now entered for v - code not shown */
if (v==5){
    printf("positive\n");
} else if (v==-5){
    printf("negative\n");
} else if (v==0){
    printf("zero\n");
} else printf("incorrect value entered\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 "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[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\n", total);
}
```

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){
    char message[80]="I love C programming";
    int i=0, count=0;
    while (message[i]!='\0'){
        if ((message[i]=='c') || (message[i]=='C')){
            count++;
        }
        i++;
    }
    printf("\n%s\n contains %d c's\n", message, count);
}
```

Re-write the above program so that all the code for determining the number of **c**'s in the string **message** is contained in a function called **c_counter()** that you should define, while the declaration and initialization of **message** and the output is still done from **main()**.

o O o