TERM	COURSE NAME	COURSE CODE	VERSION
2247	Introduction to Programming using C	IPC144	XX

Name	(write your full name here)
Student Number	(write your student number here)
Section	(write your section number here)

DATE:	___________
TIME ALLOWED:	1 hour 40 minutes
PERCENTAGE:	35%
TOTAL MARKS:	50
PROFESSOR(S):	___________

SPECIAL INSTRUCTIONS *(example)*:

1. Mobile phones, smartwatches, earbuds and other electronic devices need to be put in bags and placed under your chair or at the front or back of the room.
2. Answer all questions in the space provided on the test paper.
3. Communication is only allowed with your professor or lab monitor. There is to be no communication with other students.

This exam includes a *cover page*, plus *__six__ (6)* pages of questions.

APPROVED BY:

Question 1: Concepts (3 marks):

In two or three sentences, explain the difference between pass by address and pass by value

Question 2: Concepts (3 marks):

-> and . both allow you to access the data member of a struct. When would you use one vs the other?

Question 3: Walkthrough (20 marks)

a) fill in the blank on line 9 with the last digit of your student number (1 mark)

b) What is the exact output of the following program? 11 marks

c) Produce table(s) and arrows showing variable tracking and function call tracking-as rough work 8 marks

```
1.  #include <stdio.h>
2.  #include <string.h>
3.  #include <ctype.h>
4.  #define MAX 10
5.  void doStuff(char s[]);
6.  int main(void)
7.  {
8.
9.      int last = ________;
10.     char s1[MAX+1] = "the ";
11.     char s2[MAX+1] = "puppy ";
12.     char s3[MAX+1] = "kitty ";
13.     char s4[MAX+1] = "saw ";
14.     char s5[MAX+1] = "today ";
15.     char s6[MAX+1] = "yesterday ";
16.
17.     char final[MAX*10+1] = "";
18.     char temp[MAX*10+1]= "";
19.
20.     strcat(final, s1);
21.     printf("1: %s\n", final);
22.
23.     if(last%2 == 0){
24.             doStuff(s2);
25.             strcat(final, s2);
26.             strcat(s1,s3);
27.     }
28.     else{
29.             doStuff(s3);
30.             strcat(final, s3);
31.             strcat(s1,s2);
32.     }
33.     printf("2: %s\n", final);
34.     printf("3: %s\n", s1);
35.
36.     strcat(final, s4);
37.     printf("4: %s\n", final);
38.
39.     strcpy(temp, final);
40.
41.     printf("5: %s\n", final);
42.     printf("6: %s\n", temp);
43.
44.     if(strcmp(final,temp)==0){
45.             strcpy(final, s6);
46.             strcat(final, temp);
47.             strcat(final, s1);
48.     }
49.     else{
50.             strcpy(temp, s2);
51.             strcat(final, s5);
52.             strcat(final,temp);
53.     }
54.     printf("7: %s\n", final);
55.     printf("8: %s\n", temp);
56.
57.     return 0;
58. }
```

```
59. void doStuff(char s[]){
60.     int i;
61.     for(i = 0;s[i]!='\0';i++){
62.             if(i % 2 == 0){
63.                     s[i] = toupper(s[i]);
64.             }
65.     }
66. }
```

Programming - 30 marks total, see question for breakdown

Write the following function

```
int readRibbons(const char filename[], struct Ribbon ribbons[], int maxCapacity);
```

This function is passed the name of a file where each line of the file contains a string of data. Other than the newline character, all characters within the line are valid. You may assume that no line is longer than 1000 characters. This function will read each line in from the file and store each line into a separate element of the ribbons array. The array has a maximum capacity of maxCapacity, therefore only maxCapacity lines should be read into the array. The function returns number of lines read. the position for each Ribbon should initially be 0. The length is the length of each of the lines read in.

For example suppose your file had three lines:

```
hello world
R2D2 & C3PO
everywhere
```

You would store this information into the first 3 elements of ribbon. the position for all three would be 0. The length of the first element would be 11, the second element would be 11 and the third element would be 10. function would return 3

Question 5 (4 marks)

Write the following function

void advance(struct Ribbon* ribbon, int n);

This function is passed a pointer to a Ribbon struct and a number. It will change the **position** member of the struct Ribbon by **n**. **n** can be both positive or negative. a positive value increases **position** by **n** while a negative value decrease position by **n**. Thus if the original **position** was 5 and **n** was 2, the **position** would be 7. If the original **position** was 5 and **n** was -3, the **position** would be 2. The **position** cannot be negative, thus if **n** causes **position** to become negative, **position** is set to 0. Similarly **position** cannot be higher than the value of the **length** data member. Thus, if **n** advances **position** past **length**, position is to be set at **length**

Question 6 (4 marks)

Write the following function

```
void overUnder(int n, struct Ribbon ribbons[], int size, int* over, int* under, int* equals);
```

This function is passed:

n - a number we will be comparing against
ribbons - an array of Ribbon structs
size - the size of the ribbons array
over, under, equals - 3 pointers used to pass back the result

This function will go through the **ribbons** array and count the number of ribbons whose **length** is greater than **n**, smaller than **n** and equal to **n**. It will pass the result back through the pointers **over**, **under** and **equals** respectively

Question 7(6 marks)

Write the following function

```
int appendAtPosition(struct Ribbon* ribbon, const char str[]);
```

This function is passed a pointer to a Ribbon struct and a null terminated string. It will append **str** to **ribbon** starting at **position** within ribbon. The **length** must be updated to correctly represent the **length** of the new string. **position** is unchanged after the function call.

Suppose that your original Ribbon contained "hot air balloon", with **position** of 4 (where the 'a' in ***air*** is) and **str** was "dog". The function would turn **data** in the Ribbon to: "hot dog" and change the **length** to 7

Question 8 (5 marks):

Write the following function:

```
void printRibbon(const char filename[], struct Ribbon ribbons[], int size, int longForm);
```

This function is passed the following:
filename - name of a file where the output will go
ribbons and **size** - an array of Ribbon structs and the size of the array
longForm - holds either 0 or 1

This function will print the **ribbons** array to the file. Each line will contain the text string stored in one element of **ribbons**. If **longform** is true, it will also print a semi-colon followed by the **position**. If **longForm** is false it will only print the text string stored in **ribbons**