THE PAPUA NEW GUINEA UNIVERSITY OF TECHNOLOGY

DEPARTMENT OF MATHEMATICS AND COMPUTER SCIENCE
SECOND SEMESTER EXAMINATIONS 2021

THIRD YEAR COMPUTER SCIENCE

CS322 – C Programming

TIME ALLOWED – 3 HOURS

Information for Candidates

1. Write your name, student number, and program of study clearly on the front page of your answer booklet. Do it **now**.
2. You have 10 minutes to read this examination paper. During this time you must **NOT** write **inside** your answer booklet. You can make notes on the examination paper.
3. Scientific calculators are permitted. Other electronic devices are not permitted. Notes and headphones are not permitted.
4. Start each question on a new page. After you have finished the exam, indicate the orderin which you answered questions in the left column of the marks box on the cover of the answer booklet.
5. There are 6 questions. You should attempt all questions.
6. Total marks is 100. Marks for questions and major question parts are shown at the top of each question. Marks for minor question parts are shown at the end of each question part. The following table gives a marks breakdown per question:

Question	1	2	3	4	5	6
Total Marks	25	15	15	15	15	15

QUESTION 1 [2 + 2 + 2 + 5 + 2 + 2 + 2 + 2 + 2 + 2 + 2 = 25 marks]

Here is a program written in C with line numbers on the left.

```
1.    /*   Program 1 /*
2.    #include <stdio.h>
3.
4.    int main() {
5.       int t = 20;
6.       char a[] = "Testing 1,2,3,...";
7.       if ( t > 18 ) printf( "Output: %s \n", a );
8.       else printf( "Output: %d \n", t );
9.       puts( a+2 );
10.      return 0;
11.   }
```

(a) Line 1 is a comment. C++ uses an additional comment type.
What is it, and what its practical difference to the above traditional style used above? [2]

(b) Why is line 2 needed? [2]

(c) What is significant about the name of the function in line 4? [2]

(d) Explain (including memory implications) the syntax in line 6. [5]
Since this question part is worth 5 marks you will need 5 points!

(e) What is the significance of the **"%s"** and **"%d"** in lines 7/8? [2]

(f) What is the significance of the **"\n"** in lines 7/8? [2]

(g) When run, what would the first line displayed be, and why? [2]

(h) Suppose line 5 was replaced by `int t = 12;` Then
`printf( "Output: %d \n", t );` would display 12. If this line was replaced by
`printf( "Output: %o \n", t );` what would display, and why? [2]

(i) What would line 9 display, and why? [2]

(j) What is the use of line 10? [2]

(k) The above program uses two character types, "char" and "int". Name one other data type, and write a line of code that uses it and initializes it to a value. [2]

QUESTION 2 [4 + 4 + 2 + 2 + 2 + 1 = 15 marks]

This question involves C and/or C++ applications.

(a) What is the mechanism by which the C (or C++) source code gets converted to the final executable code (say, **a.out** on Unix or **a.exe** on Windows)? [4]

(b) Why does an application written in C (or C++) run fast (compared to say Java or Python)? [4]

(c) On **Unix** all our executables were called "a.out". Could we still have executed the programs if we had renamed "a.out" to "a.exe"? Explain [2]

(d) From a practical point of view, on **Unix** we always ran our executables using the syntax "`./a.out`" (rather than simply "`a.out`"). Why? [2]

(e) What do you understand by a "C project" (as in most IDEs), or a "C Makefile" (as traditionally used with the C command line environment)? [2]

(f) Is this statement true or false?
"C and C++ are different programming languages." [1]

QUESTION 3 [3 + 2 + 3 + 7 = 15 marks]

This question is about assembly language programming.

(a) *If a student wished to learn the fundamentals of Assembly Language Programming, it would be most advantageous if he/she first learned the fundamentals of C.*

Justify this statement. Your answer should include mention of Java or Python or both. [3]

(b) Here is a line of x86-64 assembly language code. Write a line of C code that might generate it. [2]

```
movl    $18    -12(%rbp)
```

(c) During compilation, a C program uses the CX register.
If we viewed the assembly code for the program we would see two statements involving the **system stack** that mentioned the CX register. Explain this. [3]

(d) Compilers have "sneaky" ways to perform some computational tasks.
For example, this line is performing a multiplication by a power of 2:

```
sall    $5    -8(%bp)        [sall = shift left]
```

(i) Write down the C code that might have generated this line. [2]
(ii) How does this line of assembly code perform your line of C code. [3]
(iii) Why is multiplying this way very efficient? [2]

QUESTION 4 [8 + 5 + 2 = 15 marks]

This question is about C++ object oriented programming.

(a) Here is a class declaration and function definitions:

```
class Tbase {
      int x;
      char a[1024];
   public:
      Tbase() { x = 5; }
      void setx( int x; }
      int getx() { return x; }
};
```

(i) Show the C++ code required to instantiate
* a **static** Tbase object **A**, and
* a **dynamic** Tbase object **B**. [3]

(ii) How/when would objects **A** and **B** be destroyed? (One *way for A, two for B)* ***[3]***

(iii) To display the value of "x" you might use this line of code:

```
cout << "Tbase's x vlaue is ",  << __ getx() << "\n";
```

What is the missing operator (__) for each of object types **A** and **B**? [2]

(b) **Tbase** above is to be used as a base class for a new derived class **Tderived**, **Tderived** has this declaration and function definitions:

```
class Tderived : public Tbase {
      int y;
   public:
      Tderived() { y = 10; }
      void sety( int y; }
      int getxy() { return x*y; }       ***
};
```

A **Tderived** object **C** is created.

(i) **C** will have an "**x**" member, and its value will initially be "**5**".
How does this happen? [3]

(ii) The line above labeled *** would cause an error.
Why, and what change to **Tbase** is required to correct it? [2]

(c) If a base class TB and two derived classes TD1, TD2 are appropriately set up, and the following code **causes no errors**, explain why the mentioned function "**test**" is virtual (ie, displays a polymorphic nature). [2]

```
TB * b;
TD1 * d1 = new D1;
TD2 * d2 = new D2;
int j = 0;
cin >> j;
if ( j/2 < 20 ) b = d1;
else b = d2;
cout << b -> test();
```

QUESTION 5 [4 + 6 + 5 = 15 marks]

This question involves C pointers.

(a) Explain why this C-code snippet would display "5". [4]

```
char a[50] = "CS323";
char * t = a;
while ( *t )  t++;
printf( "%d", t-a );
```

(b) In a linked list the following struct was declared, and later used as shown below. Write some notes about the purpose of the three shown lines of code in "main"? [6]
Hint: A diagram might help!

```
struct node {
   char a[100];
   struct node * next;
}

int main() {
   struct node * top = malloc( sizeof(node) );
   struct node * x = top;
         :
         :
   while ( x -> next )  x = x -> next;
         :
         :
}
```

(c) Below left is a C program. It is called from the command line this way:

```
p1.exe bob bill sue
```

The resulting output is shown on the right.

```
#include <stdio.h>

int main (int argc, char * argv[] ) {
   int j = 0;
   while ( j < argc ) {
      printf( "%s \n", argv[j] );
      j++;
   }
   return 0;
}
```

```
Output →
[hidden]
bob
bill
sue
```

(i) One line of the output has been hidden. What would it have been if not hidden? [1]
(ii) Explain the **"int argc, char * argv[]"** syntax. [2]
(iii) In the code as used above, what integer value is passed to "argc"?
Where does this integer come from? [2]

QUESTION 6 [4 + 2 + 6 + 3 = 15 marks]

This question examines GUI programming using C and the GTK graphics toolkit.

(a) At first sight, programming using Microsoft Visual Studio (perhaps using Visual C++) "feels" quite different from the method we used in class using gtk+ widgets. But essentially there are only two differences between visual and non-visual GUI programming.
(i) What are they? [2]
(ii) Explain how the two differences simplify enormously the programming task? [2]

(b) All C/C++ programs include the entry point function "main()". But there is an exception with GUI programming. Explain. [2]

(c) Explain the lines marked ***1 to ***6 in the following GTK+ code. [6]

```
void test( GtkButton *btn, gpointer ntry ) {      ***4
   gtk_entry_set_text( GTK_ENTRY (ntry) , "");
}

int main( int argc, char ** argv) {
   gtk_init( &argc, &argv );                  ***5

   GtkWidget * window = gtk_window_new(GTK_WINDOW_TOPLEVEL);***1
   gtk_window_set_default_size(GTK_WINDOW(window), 200, 200);
   gtk_window_set_title(GTK_WINDOW(window), "CS322");
   gtk_widget_show(window);

   GtkWidget * vbox = gtk_box_new( TRUE, 0);
   gtk_container_add( GTK_CONTAINER (window), vbox );        ***2
   gtk_widget_show( vbox );

   GtkWidget * entry = gtk_entry_new();
   gtk_container_add( GTK_CONTAINER (vbox), entry );
   gtk_widget_show(entry);
   gtk_entry_set_text( GTK_ENTRY (entry) , "Fill Me");

   GtkWidget * button = gtk_button_new_with_label("Click Me");
   gtk_container_add( GTK_CONTAINER (vbox), button );
   gtk_widget_show(button);
   g_signal_connect( button, "clicked", G_CALLBACK( test ), entry);  ***3

   gtk_main();       ***6
   return 0;
}
```

(d) Sketch how this program would appear initially on a computer screen. [3]

---- End of Exam ----