GLASGOW COLLEGE UESTC

Exam paper

Introductory Programming (UESTC1005)

Date: 7th Jan. 2019
Time: 09:30-11:30 am

Attempt all PARTS

Use one answer sheet for each of the questions in this exam.
Show all work on the answer sheet.
Make sure that your University of Glasgow and UESTC Student Identification Numbers are on all answer sheets.

An electronic calculator may be used provided that it does not allow text storage or display, or graphical display.

All graphs should be clearly labelled and sufficiently large so that all elements are easy to read.

The numbers in square brackets in the right-hand margin indicate the marks allotted to the part of the question against which the mark is shown. These marks are for guidance only.

Continued overleaf

Q1 (a) Write a single C statement to accomplish each of the following:

i) Read an integer from the keyboard and store the value entered in integer variable *a*. [2]

ii) If the integer variable *number* is not equal to 7, print "The variable number is not equal to 7." [2]

iii) Set variable *x* to 1. [2]

iv) Declare a function *intToFloat* that takes an integer argument, *number*, and returns a floating type result. [2]

(b) State whether each of the following is true or false. If false, explain why.

i) C considers the variables *number* and *NuMbEr* to be identical. [2]

ii) The following variable names are identical on all Standard C systems. [2]

thisisasuperduperlongname1234567
thisisasuperduperlongname1234568

iii) The expression ($x > y$ && $a < b$) is true if either $x > y$ is true or $a < b$ is true. [2]

iv) If there are fewer initializers in an initializer list than the number of elements in the array, C automatically initializes the remaining elements to the last value in the list of initializers. [2]

v) It is an error if an initializer list contains more initializers than there are elements in the array. [2]

vi) Pointers of different types may not be assigned to one another without a cast operation. [2]

Continued overleaf

Q2 (a) Consider the code fragment of Figure Q2.

i) Rewrite this code, adding appropriate comments to lines 4, 5 and 6. [3]

ii) Write down the output produced by the code fragment [5]

```
1.  int main()
2.  {
3.  int x;
4.  for ( x = 20; x >= 10; x-- ) {
5.  if ( x == 16)
6.  break;
7.  printf( "%d ", x );
8.  }
9.  return 0;
10. }
```

Figure Q2

(b) A person invests 10000 RMB in a savings account yielding 5% interest. Assuming that all interest is left on deposit in the account, we wish to calculate and print the amount of money in the account at the end of each year for 10 years. We may use the following formulae to determine these amounts:

$a = p(1 + r)^n$

where

p is the original amount invested (i.e., the principal)

r is the annual interest rate

n is the number of years

a is the amount on deposit at the end of the nth year.

Write a program by using *for* loop that performs the indicated calculation and printed output for each of the 10 years the money remains on deposit. [12]

Continued overleaf

Q3 A sensor's raw measurement is linked to a property. To get the property, each new measurement is transformed. The transformation between raw measurements and properties is shown in the following table:

Table Index	0	1	2	3	4	5	6	7	8	9	10
Raw Data	803.1	822.9	842.7	862.5	980.4	1000	1097.3	1155.4	1194	1213.2	1232.4
Property	-50	-45	-40	-35	-5	0	25	40	50	55	60

The value of the property within each interval of the table changes linearly with the raw measurement, and it is assumed that no measurement is outside the range of the table.

(a) Write a code fragment to define a C structure containing both a raw measurement and a property. [5]

(b) Given any raw measurement, write a code fragment that finds the indexes of the table bounding (i.e. above and below) that raw measurement value. [5]

(c) Calculate the value of the property associated with the raw data 1050.0. [2]

(d) Design a function which correctly calculates the property value associated with any raw measurement. [8]

Q4 Consider the code below and answer the following questions:

```
#define TIMER_10MS_MAX (0xFFFFFFFF)

typedef struct soft_timer_st {

   uint32_t tickdown;
   void* context;
   void (*cb)(void* ptr);
}soft_timer_t;

#define SOFT_TIMER_NUM (32)
soft_timer_t g_softTimer[SOFT_TIMER_NUM];

static int compare(const void* l, const void* r)
{
   soft_timer_t* a = (soft_timer_t* )l;
   soft_timer_t* b = (soft_timer_t* )r;

   if ( a->tickdown < b->tickdown ) {
      return -1;
   }
   else if ( a->tickdown == b->tickdown ) {
      return 0;
```

Continued overleaf

```
    }
    else {
        return 1;
    }
}
void st_init()
{
    memset(g_softTimer, 0xFF, sizeof(soft_timer_t)*SOFT_TIMER_NUM);
}

bool st_register(soft_timer_t* timer)
{
    if ( timer->tickdown < 1 ) return false;

    // find a place
    for (int i = 0; i < SOFT_TIMER_NUM; ++i) {
        if (g_softTimer[i].tickdown == TIMER_10MS_MAX) {
            g_softTimer[i].tickdown = timer->tickdown;
            g_softTimer[i].context = timer->context;
            g_softTimer[i].cb = timer->cb;

            // sort it
            qsort(g_softTimer, SOFT_TIMER_NUM, sizeof(soft_timer_t),
compare);
            return true;
        }
    }
    return false;
}

void st_release(soft_timer_t* timer)
{
    timer->tickdown = TIMER_10MS_MAX;
    timer->context = NULL;
    timer->cb = NULL;
}

void st_update()
{
    for (int i = 0; i < SOFT_TIMER_NUM; ++i) {
        if (g_softTimer[i].tickdown == TIMER_10MS_MAX) {
            break;
        }

        if (g_softTimer[i].tickdown-- <= 1) {
            if (g_softTimer[i].cb != NULL) {
                g_softTimer[i].cb(g_softTimer[i].context);
            }
            st_release(&g_softTimer[i]);
            // sort it
            qsort(g_softTimer, SOFT_TIMER_NUM, sizeof(soft_timer_t), compare);
        }
```

Continued overleaf

```
    }
}
void qsort (void* base, size_t num, size_t size,       int (*compar)(const
void*,const void*));
// Sorts the num elements of the array pointed to by base, each element size
bytes long, using the compar function to determine the order.
```

Figure Q4

(a) Explain in details of implementation *soft_timer_t*, *st_init*, *st_register*, *st_update*, *st_release* (2 marks for each discussion). [10]

(b) Explain why *g_softTimer[i].cb(g_softTimer[i].context)* should not call *st_register* in the current implementation. [10]

Q5 (a) A string is a sequence of zero or more characters surrounded in double quotes, write down the function prototypes for commonly-used string manipulation functions (2 marks for each correct function prototype) [10]

(b) Write a function that returns true if an input string is a palindrome. A palindrome is a word that reads the same backwards as it does forwards e.g. ABBA. [10]

End of question paper