UNIVERSITY OF EDINBURGH

FACULTY OF SCIENCE AND ENGINEERING

COMPUTER PROGRAMMING SKILLS AND CONCEPTS 1h

Tuesday, 11 June 2002,
14:00 – 15:30

Examiners
D.K. Arvind (Chair), R. Dyckhoff (External).

INSTRUCTIONS TO CANDIDATES

1. Check that the question paper contains pages 1–13. If it does not, inform the invigilator.
2. You should attempt as many questions as possible.
3. The questions in this paper vary widely in difficulty. During your first pass through the paper, you are advised not to dwell on questions to which the answer is not readily apparent.
4. Write your answer immediately below each question. Where a box or table is provided, write your answer within it. If you wish to write more in answer to a question, continue on the blank page opposite, indicating that you have done so.
5. Please write *legible* and *concise* answers.
6. The marks allocated to each part of a question are indicated in the margin. There are 100 marks in total.
7. If you are in the third or later year of your degree course write 3 here. [ ]

FOR EXAMINERS' USE

MARK [ ] (out of 100)

STUDENTS - PLEASE DO NOT WRITE ANYTHING ON THIS PAGE!

Paper No.

Question	Mark
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
Total	

1. Consider the following `if`-statement, where x,y,z are of type integer.

```
if ( x < y + z ) {
  if ( z++ ) {
    y = x + 1;
    z = y + 1;
  }
  else {
    x = y + 1;
    y = z + 1;
  }
  if ( z++ < 0 ){
    y++;
    z++;
  }
}
```

(a) Fill in the values that the variable `x,y,z` would have after executing the statement above, assuming their values are initially assigned as given. *(8 marks)*

Initial x	Initial y	Initial z	Final x	Final y	Final z
-2	1	-2			
0	0	-2			
0	2	0			
-10	-4	-5			

(b) What is the output when executing the following statements, where `a,b,c` are integer variables with the values `a=5, b=-5, c=-1;` ?

```
if (c - 1 == a + b) {
  if (c + 1) {
    a += c;
    b %= c;
  }
}
else {
  c += a++;
  b *= a;
}

printf( "a=%d,\nb=%d,\nc=%d\n", a, b, c );
```

(2 marks)

2. (a) Below is a template for character-by-character processing of an input file. Complete the program by filling in the boxes so that the program counts the number of upper case letters (using the variable `countu`) and the number of lower case letters (using the variable `countl`) and then prints out the values at the end. [*Recall that the functions* `isupper` *and* `islower` *test whether a character is upper or lower case, respectively.*]

```
int main(void) {
    int c, countu, countl;
```

$\boxed{}$ *(1 mark)*

```
    while ((c = getchar()) != EOF) {
```

$\boxed{}$ *(3 marks)*

```
    }
```

$\boxed{}$ *(1 marks)*

```
    return EXIT_SUCCESS;
}
```

(b) Below is a template for character-by-character processing of an input file. Complete the program by filling in the boxes so that the program copies the input to the output but with all occurrences of the sequence 'ez' (lower case only) replaced by 'es'.

```
int main(void) {
    int c, prev = 0;
    while ((c = getchar()) != EOF) {
```

$\boxed{}$ *(5 marks)*

```
    }
    return EXIT_SUCCESS;
}
```

3. Consider the following program fragment.

```
int m, n, sumd;
m = 28;
sumd = 0;

for( n = 1; n < m; n++ ){
  if ( m % n == 0 ) { sumd += n ; }
  }
if ( sumd == m )
  { printf( "%d is perfect\n", m ); }
else { printf( "%d is not perfect\n", m ); }
```

(a) What is the output when this code is executed?

(3 marks)

(b) Supply an initial value for `m` that will cause the body of the `for` loop never to be entered.

(2 marks)

(c) Explain why the loop will always terminate, whatever initial values are given for `m` and `sumd`.

(3 marks)

(d) Explain without mentioning the program loop what is the property of `m` called "perfect" that is computed by this code.

(2 marks)

4. Write the output of the following program in the box provided.

```
void function1(int i, int j) {
   printf("i = %d, j = %d\n", i, j);
}

int function2(int *i, int *j) {
int k;
   printf("(*i - *j) = %d\n", (*i - *j));
   k = *i;
   *i = *j;
   *j = k;
   return (*i - *j);
}

int main(void) {
   int x, y, *ax, a, b, p, q, r;

   x = 1234;
   ax = &x;
   y = *ax;
   printf("x = %d, y = %d\n", x, y);

   a = 123;
   b = 321;
   function1(a++, ++b);
   printf("a = %d, b = %d\n", a, b);

   p = 99;
   q = 100;
   r = function2(&p, &q);
   printf("p = %d, q = %d, r = %d\n", p, q, r);

   return EXIT_SUCCESS;
}
```

(10 marks)

5. For each code fragment below, write a tick in the left box if the code terminates, and a cross otherwise. In the cases that terminate, write the final value of `n` in the centre box, and of `m` in the right-hand box. In all cases assume a declaration `float n, m;` is given.

(a)
```
n = 10.0; m = 0.0;
while ( !(n < m)) {
    ++n;
}
```

terminates? ☐ value of `n`? ☐ value of `m`? ☐ *(2 marks)*

(b)
```
n = 0.5; m = 0.22;
while (n < m) {
    n += 0.1;
    m = m - n;
}
```

terminates? ☐ value of `n`? ☐ value of `m`? ☐ *(2 marks)*

(c)
```
n = 0.5; m = 2.2;
while (n < m) {
    m = m * n;
}
```

terminates? ☐ value of `n`? ☐ value of `m`? ☐ *(2 marks)*

(d)
```
n = 1.0; m = 0.1;
while (n != m) {
    m = n * m;
    n = 10 * m;
}
```

terminates? ☐ value of `n`? ☐ value of `m`? ☐ *(2 marks)*

(e)
```
n = 1; m = 16.0;
while (n < m) {
    m = m / 2;
    n = n * 2;
}
```

terminates? ☐ value of `n`? ☐ value of `m`? ☐ *(2 marks)*

6. Write the output of the following program in the box underneath it.

```
#define N 6

int main(void) {
   char name1[N], *name2, name3[N][N];
   int i, j;

   strcpy(name1, "Bill");
   name2 = name1;
   printf("name1 = %s, name2 = %s\n", name1, name2);

   strcpy(name1, "Gates");
   printf("name1 = %s, name2 = %s\n", name1, name2);

   strcpy(name2, "XBox");
   printf("name1 = %s, name2 = %s\n", name1, name2);

   for (i=0; i<N; i++) {
      for (j=0; j<i; j++) {
         name3[i][j] = '$';
      }
      name3[i][i] = '\0';
      printf("%s\n", name3[i]);
   }

   return EXIT_SUCCESS;
}
```

(10 marks)

7. The question concerns the use of a C data-type used to record information about the state of a game played on an $n \times n$ board. Each square is marked as one of **B, X, Y**, indicating blank, marked by player X or marked by player Y, respectively. The state of the game is given by a marked board, together with the board size and an indication of which player should play next.

   Some initial code is provided as shown.

```
#define MAXDIM  100    /* max size of the board */

typedef enum {B, X, Y} Square_t;
typedef enum {Player_X, Player_Y} Player_t;

typedef struct { int x, y; } Point_t;
typedef Square_t Board_t[MAXDIM][MAXDIM];
                      /* Representation of the board */

typedef struct {
  Board_t theBoard;
  Player_t toPlay;
  int size;
} Game_t;

/* global variable  */
Game_t  theGame;
```

   (a) Write a procedure to initialise a game played on an $n \times n$ board, by setting the board size, marking the appropriate squares as blank, and setting player X to play.

```
int initGame(int s)
{

}
```

   (4 marks)

P.T.O.

(b) Suppose we want to check a proposed move to see if it is legal, and to update the game if so.

First write two short procedures as follows:

```
Square_t hisSquare( Player_t p )
  /* return square X for Player_X, and Y for player_Y /*

Player_t nextPlayer( Player_t p )
  /* return opposite player  */
```

(2 marks)

Now define a *play check* procedure that takes a player and a point, and updates the game if everything is legal (it is the player's turn to move, the point is on the board and is blank). The return value should be 1 if the move is legal ond 0 otherwise.

```
int playCheck( Player_t P, Point_t XY )
{
```

(4 marks)

```
}
```

8. Give the type and value of each of the following C expressions, e.g. the `1+2` has type `int` and value `3`. [*On the final page of this paper you will find a table that gives the precedence and associativity of the various operators in C.*]

(a) `10/3`

type ☐ value ☐ *(1 mark)*

(b) `10%3`

type ☐ value ☐ *(1 mark)*

(c) `10.0/3.0`

type ☐ value ☐ *(1 mark)*

(d) `2*3.0`

type ☐ value ☐ *(1 mark)*

(e) `2*3/4*5`

type ☐ value ☐ *(1 mark)*

(f) `'A'-'B'`

type ☐ value ☐ *(1 mark)*

(g) `"A-B"`

type ☐ value ☐ *(1 mark)*

(h) `strcmp("Bill", "Gates")`

type ☐ value ☐ *(1 mark)*

(i) `1 > 2 || 3 < 4`

type ☐ value ☐ *(1 mark)*

(j) `sqrt(49.0)`

type ☐ value ☐ *(1 mark)*

9. (a) Draw a diagram, in the box provided, of the data structure which is created by the following program:

```
struct node {int data; struct node *next; };
typedef struct node list;

void main() {
   list n1, n2, n3;

   n1.data = 123;  n1.next = &n2;
   n2.data = 456;  n2.next = &n3;
   n3.data = 789;  n3.next = NULL;
}
```

(4 marks)

(b) Draw a diagram of the data structure created by the following program in the first box overleaf. Write the output from the program in the second box overleaf.

```
struct node {int data; struct node *next; };
typedef struct node list;

list *push(int data, list *old) {
  list *new;
  new = malloc(sizeof(list));
  new->data = data;
  new->next = old;
  return new;
}

void print_list(list *l) {
  while (l != NULL) {
    printf("%d\n", l->data);
    l = l->next;
  }
}

int main(void) {
   list *l = NULL;
   int i;

   for (i=1; i<5; i++) l = push(i*i, l);
   print_list(l);
}
```

(3 marks)

(3 marks)

10. Consider a binary search procedure, which searches an integer array for a given key, assuming the elements of the array are in ascending order.

```
Bool_t BinarySearch(int n, int a[], int searchKey)
/*  Returns TRUE iff searchKey is contained in the array, i.e.,
 *  there exists an index i with 0 <= i < n and a[i] == searchKey.
 */
{
   int i, j, m;
   i = 0;
   j = n - 1;

   while (i < j) {
      m = (i + j)/2;
      if (searchKey <= a[m]) {
         j = m;
      } else {
         i = m + 1;
      }
   }
   return a[i] == searchKey;
}
```

(a) Suppose `n` = 8, and the array `a` contains the values

`a[0],a[1],...  = 0, 3, 3, 7, 9, 9, 13, 17, ...`

Write down the values taken by `i` and `j` at successive iterations of the `while`-loop, when `searchKey` has the value 10. *(4 marks)*

Iteration	value of `i`	value of `j`
0		
1		
2		
3		

(b) How many times do we need to run the loop on an array storing n values as above in order to check if a search key is present or not?

(2 marks)

(c) What is the number of comparisons of items (*i.e.* calls to the `<=` check) needed to check if a search key is present or not in an array of length n?

(2 marks)

(d) What is the most number of comparisons of items needed to check if a search key is present or not in an array of length n using a *linear* search procedure?

(2 marks)

Appendix: Precedence and Associativities of some C Operators

Precedence decreases line by line from the top to the bottom of the table. Operators on the same line have the same precedence.

Operators	Associativity
`!` `+` (unary) `-` (unary)	right to left
`*` `/` `%`	left to right
`+` (binary) `-` (binary)	left to right
`<` `<=` `>` `>=`	left to right
`==` `!=`	left to right
`&&`	left to right
`||`	left to right
`=` `+=` `-=` `*=` `/=` `%=`	right to left