UNIVERSITY OF TORONTO
FACULTY OF APPLIED SCIENCE AND ENGINEERING

APS 105 — Computer Fundamentals
Midterm Examination
October 27, 2009
12:20 p.m. – 1:50 p.m.

Examiners: J. Anderson, T. Fairgrieve, H. Ghaderi, B. Li

Exam Type A: This is a "closed book" examination; no aids are permitted.

Calculator Type 4: No calculators and other electronic devices are allowed.

All questions are to be answered on the examination paper. If the space provided for a question is insufficient, you may use the last page to complete your answer. If you use the last page, please direct the marker to that page and indicate clearly on that page which question(s) you are answering there.

You must use the C programming language to answer programming questions.

The examination has 11 pages, including this one.

Circle your lecture section (**one mark deduction** if you do not correctly indicate your section):

L0101	or	**L0102**	or	**L0103**	or	**L0104**	or	**L0105**
Fairgrieve		Ghaderi		Ghaderi		Anderson		Li
Monday 2 PM		Monday 9 AM		Monday 11 AM		Monday 11 AM		Monday 4 PM

Full Name: _____

Student Number: _____ ECF Login: _____

MARKS

1	2	3	4	5	6	7	8	9	10	11	12	13	Total
/4	/4	/4	/4	/4	/4	/8	/8	/12	/12	/12	/12	/12	/100

Question 1 [4 Marks]

Write a single C statement that declares an `int` type variable named `dice`, and initializes it to be an odd random integer between -200 and 200.
Solution:

```
int dice = rand() % 200 * 2 - 199;
```

Question 2 [4 Marks]

Write a single C statement that declares an `int` type variable named `position`, and initializes it to be the position of a given lower-case letter in the English alphabet. The given lower-case English letter is stored in a `char` type variable named `c`, that has already been declared and initialized. For example, the position of the letter `'a'` is 1, and the position of `'z'` is 26.
Solution:

```
int position = c - 'a' + 1;
```

Question 3 [4 Marks]

Write a single C statement that declares a `bool` variable named `div` and assigns `true` to `div` if and only if the value stored in the `int` variable named `i` is exactly divisible by 5 or 7. Assume that variable `i` has been declared and initialized.
Solution:

```
bool div = !(i % 5) || !(i % 7);
```

Question 4 [4 Marks]

Suppose that `i` is an `int` variable with value -5, and `j` is an `int` variable with value 3. Which of the following relational expressions evaluate to `true`?

Solution:

Expression	Answer
`(i < j)`	true
`((i > j) || !(i > j))`	true
`(i + 8 < j)`	false
`((i > -5) || (j < 2))`	false

Question 5 [4 Marks]

If `i` is a type `int` variable, and `p` and `q` are type pointer to `int` variables, which of the following assignment statements are legal and which are illegal?

Solution:

Statements	Answer
`p = i;`	illegal
`*p = i;`	legal
`*q = &i;`	illegal
`p = q;`	legal

Question 6 [4 Marks]

Rewrite the following statements so that a `do` loop is used instead of a `while` loop. The results from executing the statements must remain the same.

```
int sum = 0, num;
scanf("%d", &num);
while (num >= 0)
{
  sum += num;
  scanf("%d", &num);
}
```

Solution:

```
int sum = 0, num;
do
{
  scanf("%d", &num);
  if (num >= 0)
  {
    sum += num;
    scanf("%d", &num);
  }
} while (num >= 0);
```

Question 7 [8 Marks]

Given the following C program:

```
#include <stdio.h>

int main(void)
{
  int i, j, n = 6;
  char a = ' ';
  for (i = 1; i <= n; i++)
  {
      printf("*");
      if (i == n)
      {
          a = '*';
      }
      for (j = 2; j < i; j++)
      {
          printf("%c", a);
      }
      if (i != 1)
      {
          printf("*");
      }
      printf("\n");
  }

  return 0;
}
```

What is the output from an execution of this C program?

Solution:

```
*
**
* *
*  *
*   *
******
```

Question 8 [8 Marks]

Given the following C program:

```
#include <stdio.h>

int main(void)
{

  const int N = 6;
  int i, j;

  for (i = 1; i <= N + 1; i++)
  {
    for (j = 1; j <= N; j++)
    {
      printf("%c", 'A' + ((i + j - 2) % N));
    }
    printf("\n");
  }
  return 0;
}
```

What is the output from an execution of this C program?

Solution:

```
ABCDEF
BCDEFA
CDEFAB
DEFABC
EFABCD
FABCDE
ABCDEF
```

Question 9 [12 Marks]

Write a character-valued function named `convertToGrade` that has a single type `int` parameter named `mark`. The function should return the letter that corresponds to the given `mark` according to the following table:

Mark	Letter
0 – 49	F
50 – 59	D
60 – 69	C
70 – 79	B
80 – 100	A
Others	X

Solution:

```
char convertToGrade(int mark)
{
   char result;
   if (0 <= mark && mark <= 49)
       result = 'F';
   else if (50 <= mark && mark <= 59)
       result = 'D';
   else if (60 <= mark && mark <= 69)
       result = 'C';
   else if (70 <= mark && mark <= 79)
       result = 'B';
   else if (80 <= mark && mark <= 100)
       result = 'A';
   else
       result = 'X';

   return result;
}
```

Question 10 [12 Marks]

Write a complete C program that first requests and reads two positive integers. The program should then continue to read integers, one at a time, as long as the sum total of all input integers is less than 17 and no greater than 21. Once the program has finished reading, it should print either the total, if it is 21 or less, or the words `"I lose"` if the total is over 21. You may assume that the range of integers entered by the user is between 1 and 11 (inclusive).

Here is an example output of the C program:

```
First number? 5 [enter]
Second number? 1 [enter]
Next number? 4 [enter]
Next number? 4 [enter]
Next number? 5 [enter]
Final total: 19
```

Here is another example:

```
First number? 11 [enter]
Second number? 11 [enter]
I lose
```

Solution:

```
int main(void)
{
  int total, next;

  printf("First number? ");
  scanf("%d", &total);
  printf("Second number? ");
  scanf("%d", &next);
  total += next;

  while (total < 17)
  {
    printf("Next number? ");
    scanf("%d", &next);
    total += next;
  }

  if (total <= 21)
    printf("Final total: %d\n", total);
  else
    printf("I lose\n");
}
```

Question 11 [12 Marks]

Write a complete C function named `isSymmetric`, the prototype of which is given below, that returns the `bool` value `true` if the elements of an array of integers named `values` of size `n` (specified as a parameter) are symmetric around the middle. If the array elements are not symmetric, the function should return `false`. Both the array and its size are specified as parameters.

An array `values` of size `n` is symmetric if `values[0]` is equal to `values[n-1]`, `values[1]` is equal to `values[n-2]`, and so on.

For example, the elements in the following array are symmetric:
`int list1[] = {1, 4, 5, 4, 1};`
The elements in the following array are not symmetric:
`int list2[] = {4, 6, 8, 9, 9, 8, 5, 4};`

Solution:

```
bool isSymmetric(int values[], int n)
{
  bool returnValue = true;

  int i = 0;
  int j = size - 1;

  while (returnValue && (i < j))
  {
    if (values[i] != values[j])
      returnValue = false;
    i++;
    j--;
  }
  return returnValue;
}
```

Question 12 [12 Marks]

Write a complete C program that reads an integer `n` and computes and prints the term a_n where $a_1 = 1$, $a_2 = 2$, $a_3 = 3$, and for $k > 3 : a_k = a_{k-2} + 2 \times a_{k-3}$. Assume `n` > 0. Below is the output from a sample run of the program:

```
Enter n: 5 [enter]
a_5 is 7
```

Note: In your solution, you are not allowed to use arrays, loops within loops or recursion, otherwise you will receive a grade of 0 for this question.

Solution:

```
int main(void)
{
 int a_nMinus3 = 1, a_nMinus2= 2, a_nMinus1 = 3, a_n, n;
 printf("Enter n: ");
 scanf("%d", &n);
 if (n <= 3)
 {
    printf("a_%d is %d\n", n, n);
    return 0;
 }
 for (int i = 3; i < n; i++){
    a_n = a_nMinus2 + 2 * a_nMinus3;
    a_nMinus3 = a_nMinus2;
    a_nMinus2 = a_nMinus1;
    a_nMinus1 = a_n;
 }
 printf("a_%d is %d\n", n, a_n);
 return 0;
}
```

Question 13 [12 Marks]

Assume that you are are given the following C function, which has an array of type `double` and two integers as parameters. The two `int` parameters represent array indices. The function returns the sum of the array elements between the two indices (inclusive).

```
double sumRange(double nums[], int i, int j)
{
  double sum = 0;
  for (int idx = i; idx <= j; idx++)
    sum += nums[idx];
  return sum;
}
```

Write a function named `printPartitionAverages`, whose prototype is given below, that receives an array of type `double` (named `numbers`), the length of the array (named `len`), and a partition size named `size`. The function first computes and prints the number of partitions (or subdivisions) of size `size` of the given array. And then it prints the average value for each partition. For example, for the following `len =` 6 array:

```
double x[] = {1.0, 3.0, 5.5, 6.5, 0.0, 7.0}
```

and a partition size of `size =` 2, there will be 3 partitions `{1.0, 3.0}`, `{5.5, 6.5}`, and `{0.0, 7.0}`. A call to `printPartitionAverages(x, 6, 2)` should produce the following output:

```
There are 3 partitions of size 2.
The average of partition 1 is 2.000000
The average of partition 2 is 6.000000
The average of partition 3 is 3.500000
```

Note: Assume the length of the array is a multiple of the partition size. Your function should call the `sumRange` function with appropriate arguments.

Solution:

```
void printPartitionAverages(double numbers[], int len, int size)
{
 printf("There are %d partitions of size %d\n", len / size, size);
 for (int i = 0; i < len / size; i++)
 {
    double sum = sumRange(numbers, i * size, (i + 1) * size - 1);
    printf("The average of partition %d is %f\n", i + 1, sum / size);
 }
}
```

This page has been left blank intentionally. You may use it for answers to any questions.