Carleton University
Department of Systems and Computer Engineering
SYSC 2006 B - Foundations of Imperative Programming - Fall 2013

Midterm Exam - October 17, 2013

Instructions

1. Exam questions will not be explained, and no hints will be given. If you think something is unclear or ambiguous, make a reasonable assumption (one that does not contradict the question), write it at the start of your solution, and answer the question.
2. You do not need to write header comments or inline comments for your functions. You do not need to copy any code from the question paper to your answer booklet.
3. Turn in all pages of this question paper with your answer booklet.

Question 1 [20 marks]

In mathematics, Pascal's triangle is a triangular array of binominal coefficients. Each numbers in Pascal's triangle is obtained with the formula:

r choose c = r! / c! (r-c)!

Where *r* represents the 0-based row number and *c* represents the 0-based column number in the triangle. For example, the number in row 4 column 2 is obtained by: 4 choose 2 = 6.

Write a C program (a `main` function) that prompts the user to enter the number of rows *n*. The program then prints the first *n* rows of Pascal's triangle. An example of the output is shown below:

```
Enter the number of rows: 6
     1
    1 1
   1 2 1
  1 3 3 1
 1 4 6 4 1
1 5 10 10 5 1
```

You can assume that you have been provided with a function that calculates factorials. The function's prototype is:

```
int factorial(int n);
```

For values of `n` that are greater than or equal to 0, this function calculates and returns `n!` Your `main` function must call `factorial`. Do not write `factorial` itself.

Question 2 [30 marks]

When writing a name or a title, it is a common convention to use capital letters to start the words. This is called Title Case. The following are examples:

- Snow White And The Seven Dwarfs
- Newcastle Upon Tyne / Brighton On Sea
- The DiMaggio Line
- The Last Of The Mohicans

a) [20 marks] Write a C function `toTitleCase` that takes as input a string (char []) and converts it in-place to Title Case, i.e., capitalizes the start of every word leaving other characters intact. The prototype of the function is as follows:

```
void toTitleCase (char s[]);
```

You may refer to the ASCII table attached at the end of this question sheet to figure out how to convert characters to upper case. Do not use any of the functions in `<string.h>` in your implementation.

b) [10 marks] Write a C function `testTitleCase` that acts as a test harness to the function in (a). The prototype of the function is as follows:

```
void testTitleCase (char s[], const char t[]);
```

Where parameter `s` is the original string to convert and parameter `t` is the expected string after conversion. The function should print useful information about the test on the console, including 1) the original string `s`, 2) the string `s` after conversion, 3) the expected string `t`, and 4) whether the test is a `Success` or a `Failure` based on whether the two strings are identical. For example:

If the function is called with:

```
char s[] = "The course is easy";
testTitleCase(s, "The Course Is Easy");
```

It should print (assuming no bug in your `toTitleCase` function):

```
Original: The course is easy
Converted: The Course Is Easy
Expected: The Course Is Easy
Test: Success
```

Note: You may use the `strcmp` function from `<string.h>` in your implementation. The function has the following prototype:

```
int strcmp (const char s1[], const char s2[]);
```

and returns 0 when `s1` is identical to, <0 when `s1` is smaller than, and >0 when `s1` is bigger than `s2`.

Question 3 [25 marks]

The function `HCF` below takes two positive integers and returns their highest common factor (HCF). For example, the HCF for 14 (7 x 2) and 35 (7 x 5) is 7. A `main` function calls the `HCF` function.

```
int HCF(int num1, int num2) {
   while(num1 != num2) {
      if(num1 > num2) /* B */
         num1 -= num2;
      else
         num2 -= num1;
   }
   return num1;
}

int main(void) {
   int n1=15, n2=6, n3;
   n3 = HCF(n1, n2); /* A */
   return 0; /* C */
}
```

a) [10 mark] Using the notation presented in lectures, draw a memory diagram that shows the stack frames immediately **before** each of the statements at lines `A`, `B` and `C` execute. Since line `B` is within a loop, each time you reach it, draw **new** stack frames annotated `B1`, `B2`, ...etc. instead of updating values in previous frames.

b) [10 mark] The following is a different but equivalent implementation of function `HCF`. Draw a memory diagram similar to the one in (a) but this time using the new implementation of `HCF`.

```
int HCF(int num1, int num2) {
   int i, hcf=1;
   for(i=2; i<=num1 || i<=num2; ++i) {
      if(num1%i==0 && num2%i==0) /* B */
         hcf=i;
   }
   return hcf;
}
```

c) [5 mark] Which of the two `HCF` implementations do you think is better? Justify your answer.

Question 4 [25 marks]

The following C program manipulates some integer and pointer variables.

```
#include<stdlib.h>

int main(void) {
   int n1, n2, *p1, *p2;
   p1 = &n1;
   p2 = &n2;
   n1 = 5; /* A */
   *p2 = *p1 + 3;
   (*p1)++; /* B */
   p2 = malloc(2*sizeof(int));
   *p2 = *p1 + n2;
   *(p2+1) = (*p1)-- + 5; /* C */
   p1 = p2++;
   *(--p2) *= 2; /* D */
   p1 = &n2;
   free(p2); /* E */
   free(p1); /* F */
   p1 = p2;
   *p1 = n1 + n2; /* G */
   p2 = malloc(sizeof(int));
   *(p2+1) = 3; /* H */
   p2 = p1; /* I */
   …
}
```

a) [13 marks] Draw a memory diagram showing stack frame and heap immediately **after** the statements at lines `A`, `B`, `C`, `D` and `E` execute. Draw a new stack frame and heap for each of these lines, clearly labeling them with their corresponding line numbers. Remember to use arrows only to show pointers.

b) [3 marks] Explain the problem with the statement at line `F`.

c) [3 marks] Explain the problem with the statement at line `G`.

d) [3 marks] Explain the problem with the statement at line `H`.

e) [3 marks] Explain the problem with the statement at line `I`.

ASCII Table

Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char
0	0	0		32	20	40	[space]	64	40	100	@	96	60	140	`
1	1	1		33	21	41	!	65	41	101	A	97	61	141	a
2	2	2		34	22	42	"	66	42	102	B	98	62	142	b
3	3	3		35	23	43	#	67	43	103	C	99	63	143	c
4	4	4		36	24	44	$	68	44	104	D	100	64	144	d
5	5	5		37	25	45	%	69	45	105	E	101	65	145	e
6	6	6		38	26	46	&	70	46	106	F	102	66	146	f
7	7	7		39	27	47	'	71	47	107	G	103	67	147	g
8	8	10		40	28	50	(	72	48	110	H	104	68	150	h
9	9	11		41	29	51	)	73	49	111	I	105	69	151	i
10	A	12		42	2A	52	*	74	4A	112	J	106	6A	152	j
11	B	13		43	2B	53	+	75	4B	113	K	107	6B	153	k
12	C	14		44	2C	54	,	76	4C	114	L	108	6C	154	l
13	D	15		45	2D	55	-	77	4D	115	M	109	6D	155	m
14	E	16		46	2E	56	.	78	4E	116	N	110	6E	156	n
15	F	17		47	2F	57	/	79	4F	117	O	111	6F	157	o
16	10	20		48	30	60	0	80	50	120	P	112	70	160	p
17	11	21		49	31	61	1	81	51	121	Q	113	71	161	q
18	12	22		50	32	62	2	82	52	122	R	114	72	162	r
19	13	23		51	33	63	3	83	53	123	S	115	73	163	s
20	14	24		52	34	64	4	84	54	124	T	116	74	164	t
21	15	25		53	35	65	5	85	55	125	U	117	75	165	u
22	16	26		54	36	66	6	86	56	126	V	118	76	166	v
23	17	27		55	37	67	7	87	57	127	W	119	77	167	w
24	18	30		56	38	70	8	88	58	130	X	120	78	170	x
25	19	31		57	39	71	9	89	59	131	Y	121	79	171	y
26	1A	32		58	3A	72	:	90	5A	132	Z	122	7A	172	z
27	1B	33		59	3B	73	;	91	5B	133	[	123	7B	173	{
28	1C	34		60	3C	74	<	92	5C	134	\	124	7C	174	|
29	1D	35		61	3D	75	=	93	5D	135	]	125	7D	175	}
30	1E	36		62	3E	76	>	94	5E	136	^	126	7E	176	~
31	1F	37		63	3F	77	?	95	5F	137	_	127	7F	177	