

15-122 : Principles of Imperative Computation, Spring 2016

Written Homework C

Due: Tuesday 12th April, 2016

Name: _____

Andrew ID: _____

Section: _____

This written homework will deal with some introductory C concepts.

The assignment is due by 1:30pm on Tuesday 12th April, 2016.

- This assignment can be completed in one of two ways:
- (A) by printing this file, handwriting your answers, and scanning it, or
 - (B) by editing this file and **printing it to another PDF file**

You shall then submit your solution to Gradescope.

4pts

1. Pass by reference and arrays versus pointers in C

The following little program allocates and initializes an array of integers, then calls a function to swap two of its elements. Rewrite the function `main` in the box below to use array notation instead of pointer notation wherever possible.

```
1 #include <stdlib.h>
2 #include <stdio.h>
3 #include "lib/xalloc.h"
4 #include "lib/contracts.h"
5
6 void swap(int *x, int *y) {
7     REQUIRES(x != NULL && y != NULL);
8     int t = *x;
9     *x = *y;
10    *y = t;
11    return;
12 }
13
14 int main() {
15     int* A = xmalloc(sizeof(int) * 10);
16     for (int i = 0 ; i < 10 ; i++) {
17         ASSERT(0 <= i);
18         *(A + i) = i;
19     }
20     ASSERT(*(A+2) == 2);
21     ASSERT(*(A+4) == 4);
22     swap(A+2, A+4);
23     ASSERT(*(A+2) == 4);
24     ASSERT(*(A+4) == 2);
25
26     printf("All tests passed.\n");
27     return 0;
28 }
```

```
int main() {
```

```
}
```

2. C Program Behavior

Each of the following C programs contains one or more errors. *Briefly* explain what is conceptually wrong with each example. No credit will be given if you simply copy error messages from the compiler, the runtime system, or `valgrind`. Of course you are encouraged to use these tools to help you understand the problems.

1pt

```
(a)
1 #include <stdio.h>
2 #define DIV(X,Y) (X/Y)
3
4 int main() {
5     int c = DIV(10-1, 2+3);
6     printf("(10-1)/(2+3) is = %d\n", c);
7     return 0;
8 }
```

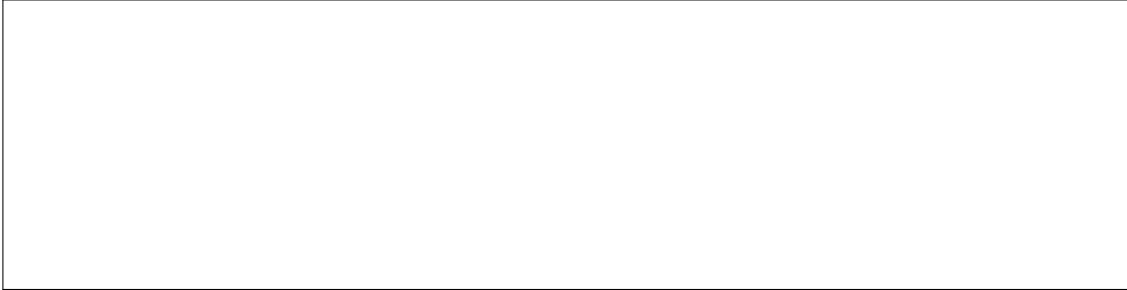
1pt

```
(b)
1 #include <stdlib.h>
2 #include "lib/xalloc.h"
3
4 int main() {
5     int *A = xmalloc(100);
6     for (int i=0; i<100; i++)
7         *(A+i) = i*i;
8     free(A);
9     return 0;
10 }
```

1pt

(c)

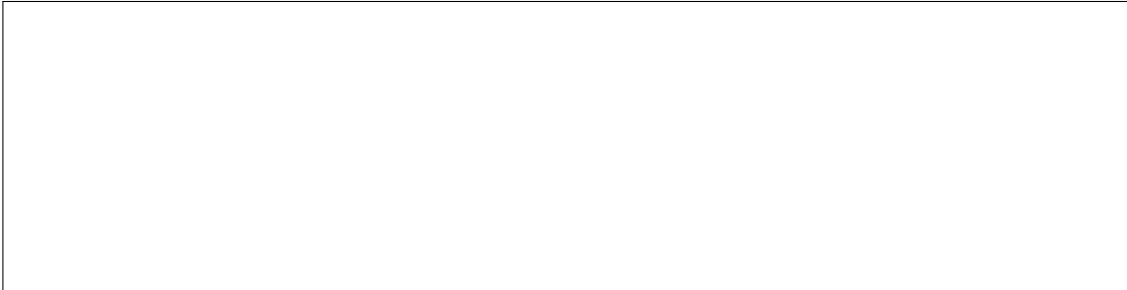
```
1 #include <stdio.h>
2 int main() {
3     char* s = "1 is the loneliest number";
4     printf("s: %s\n", s);
5     *s = '0';
6     printf("s: %s\n", s);
7     return 0;
8 }
```



1pt

(d)

```
1 #include <stdlib.h>
2 #include "lib/xalloc.h"
3 #include "lib/contracts.h"
4
5 int main() {
6     int* A = xmalloc(sizeof(int) * 10);
7     for (int i = 1 ; i < 10 ; i++) {
8         ASSERT(1 <= i);
9         *(A + i) = i;
10    }
11    free(A+1);
12    return 0;
13 }
```



1pt

```
(e)
1 #include <stdlib.h>
2 #include <stdio.h>
3 #include "lib/xalloc.h"
4 #include "lib/contracts.h"
5
6 int main() {
7     int* A = xmalloc(sizeof(int) * 10);
8     printf("Before: %d\n", A[0]);
9     for (int i = 0 ; i < 10 ; i++) {
10        ASSERT(0 <= i);
11        A[i] = i;
12    }
13    printf("After: %d\n", A[0]);
14    free(A);
15    return 0;
16 }
```

1pt

```
(f)
1 #include <stdlib.h>
2 #include <stdio.h>
3 #include "lib/xalloc.h"
4 #include "lib/contracts.h"
5
6 int main() {
7     int* A = xmalloc(sizeof(int) * 10);
8     int* B = A+3;
9     for (int i = 0 ; i < 10 ; i++) {
10        ASSERT(0 <= i);
11        A[i] = i;
12    }
13    free(A);
14    printf("B: %d\n", *B);
15    return 0;
16 }
```

1pt

```
(g)
1 #include <stdlib.h>
2 #include "lib/xalloc.h"
3
4 int main() {
5     int* A = xmalloc(sizeof(int) * 12);
6     int* B = A;
7     for (int i = 0 ; i < 12 ; i++) {
8         A[i] = i;
9     }
10    free(A);
11    for (int i = 1 ; i < 12 ; i++) {
12        B[i] = B[i] + B[i-1];
13    }
14    free(B);
15    return 0;
16 }
```

1pt

```
(h)
1 #include <stdlib.h>
2 #include <stdio.h>
3 #include "lib/xalloc.h"
4
5 int main() {
6     int* A = xmalloc(sizeof(int) * 32);
7     for (int i = 0 ; i < 32 ; i++) {
8         A[i] = i + 4;
9     }
10    int* B;
11    for (int* B = A; *B != 0; B++) {
12        printf("A[i]: %d\n", *B);
13    }
14    free(B);
15    return 0;
16 }
```

3. Integer Types

5pts

- (a) Suppose that we are working with the usual 2's complement implementation of unsigned and signed **char** (8 bits, one byte), **short** (16 bits, two bytes) and **int** (32 bits, four bytes).

We begin with the following declarations:

```
signed char the_char = -7;
unsigned char un_char_1 = 248;
unsigned char un_char_2 = 5;
int the_int = -247;
```

Fill in the table below. In the third column, always use two hex digits to represent a **char**, four hex digits to represent a **short**, and eight hex digits to represent an **int**. You might find these numbers useful: $2^8 = 256$, $2^{16} = 65536$ and $2^{32} = 4294967296$. Most, but not all, of these answers can be derived from the lecture notes. If you can't find an answer from the lecture notes, you can look at online C references or just compile some code.

C expression	Decimal value	Hexadecimal
the_char	-7	0xF9
(unsigned char) the_char	249	0xF9
(int) the_char	-7	0xFFFFFFFF9
un_char_1	248	_____
(int)(signed char)un_char_1	_____	_____
(int)(unsigned int)un_char_1	_____	_____
un_char_2	5	0x05
(int)(signed char)un_char_2	_____	_____
(int)(unsigned int)un_char_2	_____	_____
the_int	-247	_____
(unsigned int)the_int	_____	_____
(char)the_int	_____	_____
(short)the_int	_____	_____
(unsigned short)the_int	_____	_____

3pts

- (b) For this question, assume that **char** is a 1-byte signed integer type and that **unsigned int** is a 4-byte unsigned integer type.

Write the C function `pack_cui` which takes a **char** array of length 4 and packs it into a single **unsigned int**. We want the 0th character aligned at the most significant byte, and the last character aligned at the least significant byte. For example, given an array `C = {1, 2, -1, 4}`, `pack_cui(C)` should return `0x0102FF04`.

For full credit,

- Do not cast (or otherwise convert types) directly between signed and unsigned types of different sizes.
- Do not rely on the *endianness*¹ of your machine. For example, the following code is incorrect:

```
unsigned int pack_cui(char* C) { return *((unsigned int*) C); }
```

- Make sure your solution works for **char** arrays containing negative values.
- Write code that is clear and straightforward.

```
unsigned int pack_cui(char *C) {
```

```
}
```

¹“Endianness” refers to the natural storage order of bytes for a particular hardware architecture; you can read about it on Wikipedia, and don’t forget to read *Gulliver’s Travels* in your no doubt copious spare time.