HW 3

Joshua Thompson

29 JAN 2018

1. (5 points) Write a small C program that uses `sizeof()` to report the size in byte of each the types listed below. (You don't need to submit the program, just the sizes.)

   (a) `int`

   4 bytes

   (b) `char`

   1 byte

   (c) `int *`

   8 bytes

   (d) `float *`

   8 bytes

   (e) `char *`

   8 bytes

   (f) `short`

   2 bytes

   (g) `int **`

   8 bytes

   (h) `float`

   4 bytes

   (i) `double`

   8 bytes

   enumerate

   (j) (5 Points) For the sizes above, why is it that all the pointer types, even the double pointer, have the same size in bytes?

Because pointers are really just addresses. The addresses are all the same length and can either point to arrays of values or to the values themselves.

(k) (10 points) Rewrite the following C++ code in C:

```
#include <stdio>
using namespace std;

int main(int argc, char *argv[]){

 int j=10;
 int k;

 cout << "Enter a number" << endl;
 cin >> k;

 cout << "Num+10: " << k + 10 << endl;
}
```

```
#include <stdio>
using namespace std;

int main(int argc, char *argv[]){

int j=10;
  int k;

  printf("Enter a number ");
  scanf("%d", &k);
  k += j;
  printf("Num+10: %d\n", k);
  return 0;
}
```

(l) (10 points) Complete the program below to print "Go Navy" to a new file called `gonavy.txt`, "Beat Army" to a `beatarmy.txt`, and "Crash Airforce" to standard error.

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char * argv[]){

   FILE * gonavy, *beatarmy;

   //WRITE THE REST!

}
```

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char * argv[]){

   FILE * gonavy, *beatarmy;

   gonavy = fopen("gonavy.txt", "w");
   fprintf(gonavy, "Go Navy\n");
   fclose(gonavy);

   beatarmy = fopen("beatarmy.txt", "w");
   fprintf(beatarmy, "Beat Army\n");
   fclose(beatarmy);

   fprintf(stderr, "Crash Airforce\n");
   return 0;
}
```

(m) (10 points) For the following program snippet below, there are at least **four** errors. Enumerate them.

```
for( int i=0 ; i<5 , i--){
   printf(i)
}
```

-The interger i must be declared before the for loop.
-In order for the loop to loop 5 times the end statement should be ”i++” instead of ”i–”
-the printf statement should have a semicolon at the end
-The printf statement should be written like this printf(”

(n) (10 points) For the following code snippets, what is the output and explain that output? (Yes, you should program and run these program to determine the output!)

i.
```
unsigned int i = 4294967295;
printf("%d\n",i);
```

The output is -1. unsigned intergers are not supposed to return negative numbers because of how they are written. This means that the given value for i was too large so it returned -1 as an error.

ii.
```
int i = 3.1519;
printf("%d\n", i);
```

The output was 3 because the type of the variable is int. This means that it will not take any decimal values. The number was cut to 3.

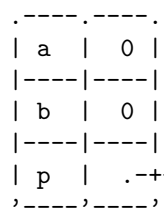
iii.
```
int i = (int) 1.5 + 2.5 + 3.5 + 4.5
printf("%d\n",i);
```

The output is 11. This is because there is a call stating that it wants 1.5 to be evaluated as int. Therefore it is shortened to just 1. Then the operation is carried out as normal which results in 11.5. 11.5 is then assigned to int variable i which shortens 11.5 to just 11.

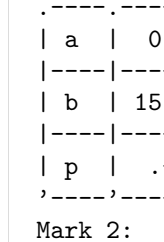
(o) (10 points) Consider the program snippet below and the memory diagram representing that programs state at MARK 0. Complete a stack diagram for each of the remaining MARKS 1-4.

```
int a=0,b=0,*p;
p = &b; /* (0) */

*p = 15; /* (1) */

a = b;

b = 25; /* (2) */

p = &a; /* (3) */

(*p)++; /* (4) */
```

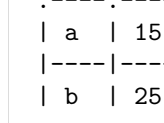
Mark 0 Diagram



Mark 1:



Mark 2:



```
|----|----|   |
| p  |  .-+---'
'----'----'
Mark 3:
.----.----.
| a  | 15 | <-.
|----|----|   |
| b  | 25 |   |
|----|----|   |
| p  |  .-+---'
'----'----'
Mark 4:
.----.----.
| a  | 16 | <-.
|----|----|   |
| b  | 25 |   |
|----|----|   |
| p  |  .-+---'
'----'----'
```

(p) (10 points) What is the values of the array after this code completes?

```
//statically declaring an array
int array[10] = {0,1,2,3,4,5,6,7,8,9};
int * p = array+3;

p[0]=2018;

//  ****ARRAY = [0,1,2,2018,4,5,6,7,8,9]****
```

(q) (10 points) You are trying to copy an array from to another, and you write the following code:

```
int a[10] = {0,1,2,3,4,5,6,7,8,9};
int b[10];

//copy from a to b
b=a;
```

i. Why is this code incorrect?

> This code is incorrect because the variable only holds the ADDRESS to the array, not the actual contents. When you put the statement b=a that only means that b will now point to the same array as a.

ii. Write a corrected code segment by replacing the offensive part of the code above to copy the values in `a` to `b`.

> int i; for(i = 0; i ¡ 10; i++) b[i] = a[i];

(r) (10 points) The program below has at least three things wrong with them. Enumerate them and write the corrected code.

```
#include <stdlib.h>
int main( int argc, char * argv[]){
  file * stream;

  stream = open("file.txt", "r");

  fprintf(stream, "Hello World");

  return 0;
}
```

-instead of ”include ¡stdlib.h¿” we need ¡stdio.h¿
-we need FILE * stream (file needs to be all caps)
- stream = open should be FILE * stream = fopen(”file.txt”, ”r”)
-the fopen command also is in read mode when it should be in write

CODE
int main(int argc, char * argv[])
FILE * stream;
stream = fopen(”file.txt”, ”w”);
fprintf(stream, ”Hello World”);
fclose(stream);
return 0;