HW 3: C Programming #1 Name Umberto Fontana

IC221, Spring AY23

100 points total

1. (18 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 write the sizes.) Note: you should run the program on a lab machine or a VM (not WSL). You can also ssh into csmidn. See sizes.c

|  |  |
| --- | --- |
| int | 4 |
| char | 1 |
| int \* | 8 (64 bytes system) |
| float \* | 8 |
| char \* | 8 |
| short | 2 |
| int \*\* | 8 |
| float | 4 |
| double | 8 |

2. (3 Points) For the sizes above, why is it that all the pointer types, even the double pointer, have the same size in bytes?

|  |
| --- |
| Because a pointer value contains an address in memory – it doesn’t matter the type of variable that the address is pointing to. |

3. (11 points) Rewrite the following C++ code into C:

|  |  |
| --- | --- |
| #include <stdio>  using namespace std;  int main(){  int j=10;  int k;  cout << "Enter a number" << endl;  cin >> k;  cout << "Num+10: " << k + 10 << endl;  } | #include <stdio.h>  int main() {  int k;  printf(“Enter a number:\n”);  scanf(“%d”, &k);  printf(“Num + 10: &d\n”, k+10);  } |

4. (15 points) Complete the program below to do these things:

* Write "Go Navy" to a new file called gonavy.txt
* Write "Beat Army" to a new file called beatarmy.txt
* Write "Crash Airforce" to standard error.
* Close the two text files after writing to them.

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  int main(int argc, char \* argv[]){  FILE \* gonavy = fopen(“gonavy.txt”, “w”);  FILE \* beatarmy = fopen(“beatarmy.txt”, “w”);  fprintf(gonavy, “Go Navy”);  fprintf(beatarmy, “Beat Army”);  fclose(gonavy);  fclose(beatarmy);  fprintf(stderr, “Crash Airforce);  } |

5. (8 points) For the following C program snippet below, there are at least four errors. List as many as you can.

|  |
| --- |
| for(int i=0 ; i < 5 , i--){  printf(i)  } |
| The “,” instead of the “;” in the for declaration (1st error).  (i--) instead of (i++) (2nd error).  The printf statement should be printf(“%d”, i); (3rd and 4th error). |

6. (15 points) For the following code snippets, say what is the output, and explain why.   
(Hint: you can actually run this code to see the output!)

|  |  |
| --- | --- |
| unsigned int i = 4294967295;  printf("%d\n", i); | Output: -1  Explanation: %d expects a signed integer. %u is the format specifier for an unsigned integer. |
| int i = 3.1519;  printf("%d\n", i); | Output: 3  Explanation: %d expects a signed integer, so it truncates the float. |
| int i = (int) 1.5 + 2.5 + 3.5 + 4.5;  printf("%d\n", i); | Output: 11  Explanation: The int typecasting only applies to 1.5, truncating it to 1, so the result of the operation is 11.5 that gets truncated by the format specifier as above. |

7. (12 points) Consider the program snippet below and the memory diagram representing that programs state at MARK 0. Complete a memory diagram for each of the remaining MARKS 1-4 by updating the values and the pointer p.

|  |  |
| --- | --- |
| int a=0, b=0, \*p;  p = &b; /\* (0) \*/  \*p = 15; /\* (1) \*/  a = b;  b = 25; /\* (2) \*/  p = &a; /\* (3) \*/  (\*p)++; /\* (4) \*/ | Mark 0  .----.----.  | a | 0 |  |----|----|  | b | 0 | <-.  |----|----| |  | p | .-+---'  '----'----' |
| Mark 1  .----.----.  | a | 0 |  |----|----|  | b | 15 | <-.  |----|----| |  | p | .-+---'  '----'----' | Mark 2  .----.----.  | a | 15 |  |----|----|  | b | 25 | <-.  |----|----| |  | p | .-+---'  '----'----' |
| Mark 3  .----.----.  | a | 15 |<- .  |----|----| |  | b | 25 | .  |----|----| |  | p | .-+---'  '----'----' | Mark 4  .----.----.  | a | 16 |<- .  |----|----| |  | b | 25 | .  |----|----| |  | p | .-+---'  '----'----' |

8. (8 points) What are the values in array after the 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]=1992;  //<--- Array values here: |
| Answer: {0,1,2,1992,4,5,6,7,8,9} |

9. (6 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;

Why is this code incorrect?

|  |
| --- |
| Array references are constant: they can’t be reassigned like pointers. Array references reflect fixed locations allocated in memory; they cannot be dynamically reassigned. |

10. (4 points) Write a corrected code segment to copy all the values from array a into array b.

|  |
| --- |
| for (int i = 0; i < 10; i++) {    b[i] = a[i];  } |