Student Name: ______________________________

Student ID: ______________________________

1	2	3	4	5	Total

Marmara University – Faculty of Engineering

Computer Engineering

Spring 2019-2020

CSE 2138 – Systems Programming

Online Homework #2

29 June 2020

I hereby swear that the work done on this online homework is totally my own; and on my honor, I have neither given nor received any unauthorized and/or inappropriate assistance for this homework. I understand that by the school code, violation of these principles will lead to a zero grade and is subject to harsh discipline issues.

Full Name: *Signature:*

Solution and Submission Notes:

1. Book, slides and notes are open.
2. Internet usage is not allowed; however, computers and cell phones may be used only for viewing course materials provided for this course.
3. This online homework consists of 5 questions and 5 pages.
4. Please write your answers clearly and neatly to the appropriate place.
5. **Show all your work.**
6. Write your student ID, first name and last name, and total number of pages you used on top of each solution page.
7. If you have not printed this PDF file, write the sentences in the frame above and the date, in your handwriting, on top of the first page of your answer sheet, and sign it.
8. Write your solutions in your handwriting on blank A4 sheets.
9. Scan all the solution pages to a single PDF file, name it as "FirstName_LastName_ID.pdf".

Upload your PDF file to http://ues.marmara.edu.tr before deadline.

1. **(16 pts)** Consider the following C code, where M and N are constants declared with `#define` and the assembly code generated for it on a 64-bit machine:

```
#define M _________;
#define N _________;

int array1[M][N];
int array2[N][M];

int copy(int i, int j){
   array1[i][j] = array2[j][i];
}
```

```
copy:
      pushq      %rbp
      movq       %rsp, %rbp
      movslq     %esi, %rax
      movslq     %edi, %rcx
      leaq       (%rax,%rax,8), %rdx
      shlq       $3, %rdx
      movl       (%rdx,%rcx,4), %edx
      imulq      $84, %rcx, %rcx
      movl       %edx, (%rcx,%rax,4)
      popq       %rbp
      retq
```

What are the values of M and N? M =   N =

<u>Explain your answer to get points.</u> <u>**Show all your work!**</u>

2. **(20 pts)** Consider the following C code and the assembly code generated for it. Fill in the blanks with the appropriate expressions. (Note: you may only use symbolic variables <u>x, y, z, i, and result</u>, from the source code in your expressions below – do not use register names.)

<u>**Show all your work!**</u>

```
foo:
  pushq     %rbp
  movq      %rsp, %rbp
  movl      %edi, %eax
  testl     %esi, %esi
  jle       LBB0_4
  movl      $1, %ecx
LBB0_2:
  movl      %ecx, %eax
  leal      (%rax,%rax), %ecx
  cmpl      %esi, %ecx
  jle       LBB0_2
  leal      (%rax,%rax,2), %eax
  addl      %edx, %eax
LBB0_4:
  popq      %rbp
  retq
```

```
int foo(int x, int y, int z){

   int i, result;

   result = ________;

   for (i = ________; ________; ________) {

      result = ___________;

   }

   return result;

}
```

3. **(17 pts)** Consider the following function for computing the dot product of two vectors.

```
long product(long v1[], long v2[], int size){
      long prod = 0;
      int i;
      for (i=0; i<=size-1; i++) {
            prod = prod + (v1[i] * v2[i]);
      }
      return prod;
}
```

Rewrite the code for function `product` that operates just the same and <u>unrolls the loop by a factor of 4.</u>

<u>Show all your work!</u>

4. **(27)** Assume that, your machine has the following properties:
    - a 16KB L1 data cache and a 1MB L2 cache that are both cold initially,
    - cache block size is 128B,
    - if an L1 miss occurs, access to L2 cache takes 10 cycles
    - if both L1 and L2 misses occur, access to memory takes 100 cycles
    - integer size is 4B

Considering the following piece of code, calculate the total number of misses and total miss cycles for the given values of N.

```
int a[N], k;
...
for (int i=0; i<N; i++)
   for (int j=0; j<N; j++)
      k = a[j];
```

a) For N=2^{12}: **total number of L1 misses =**
total number of L2 misses =
total miss cycles =

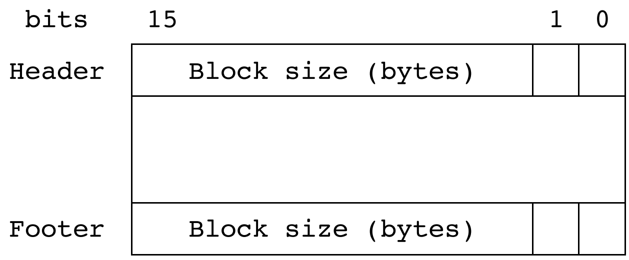
b) For N=2^{17}: **total number of L1 misses =**
total number of L2 misses =
total miss cycles =

c) For N=2^{20}: **total number of L1 misses =**
total number of L2 misses =
total miss cycles =

<u>Show all your work!</u>

5. **(30)** Consider a memory allocator with the following properties:

- Allocations are 4-byte aligned, for each memory block, either allocated or free
- Allocator uses an implicit free list
- Allocator uses immediate coalescing, that is, adjacent free blocks are merged immediately each time a block is freed
- To find a free block, first fit method is used
- The allocated and free blocks both have header and footer
- The header and footer use:
  - the higher order 14 bits to record block size (block size includes the header and footer)
  - the lower order 2 bits to record allocation information:
    - bit 0 indicates the use of current block: 1 for allocated, 0 for free
    - bit 1 indicates the use of previous adjacent block: 1 for allocated, 0 for free.



If the heap starts at memory address 0xb000, show the contents of heap after each of the following requests. Note that the addresses grow from top down.

i. Draw the heap
ii. Indicate the memory location that p1, p2, and p3 points at.
iii. Indicate each memory block either allocated or free.
iv. Indicate the blocks that contain header and footer.
v. Give the contents of header and footer for each block (in hexadecimal). Don't forget to indicate the change in header and footer when there is a need.

a) `p1 = malloc(3)`
b) `p2 = malloc(5)`
c) `free(p1)`
d) `p3 = malloc(9)`
e) `free(p2)`
f) `p4 = malloc(4)`

Show all your work and briefly explain your answer!

Your heap will be like the following, and you will draw the heap 5 times (i.e., for each request). You can insert more rows as you need.

Address	**Content**	**Header/Footer ?**	**is pointed ?**
0xb000			
0xb002			
0xb004			
0xb006			
0xb008			
0xb00a			
0xb00c			

Here is an example that shows what you will include in your answer. **Note that this example is for a totally different allocator.**

Address	**Content**	**Header/Footer ?**	**is pointed ?**
0x400AFFC	0x00000013	Header	
0x400B000	Allocated		p1 points at this memory location
0x400b004	Allocated		
0x400b008	0x00000013	Footer	
0x400b00C	0x00000012	Header	
0x400b010	Free		
0x400b014	Free		
0x400b018	0x00000012	Footer	