COMP201 Spring 2021 Final Exam
Duration: **120 minutes**

Student ID: |__|__|__|__|__|__|__|__|

Lab Section: |__|__|

First Name(s): ______________________ **Last Name:** ______________________

*Do **not** turn this page until you are told to do so.*
In the meantime, please read the instructions below.

Good Luck!

This exam contains **7 multi-part questions** and you have **120 minutes** to earn 100 marks.

- When the exam begins, please write your student ID, name and lecture section on top of this page, and sign the academic integrity code given below.
- Check that the exam booklet contains a total of **15 pages**, including this one.
- This exam is an **open book and notes exam**.
- Show all work, as partial credit will be given since you will be graded not only on the correctness and efficiency of your answers, but also on your clarity that you express it. Be neat.

HC:	_______/ 3
# 1:	_______/ 16
# 2:	_______/ 15
# 3:	_______/ 12
# 4:	_______/ 12
# 5:	_______/ 14
# 6:	_______/ 18
# 7:	_______/ 10
TOTAL:	_______/ 100

I hereby declare that I have completed this exam individually, without support from anyone else.

I hereby accept that only the below listed sources are approved to be used during this open-source quiz:
(i) Coursebook,
(ii) All material that is made available to students via Blackboard for this course, and
(iii) Notes taken by me during lectures.

I have not used, accessed or taken any unpermitted information from any other source. Hence, all effort belongs to me.

Signature: ________________________

Question 1A. Calling Functions in Assembly [16 POINTS]

In the following, you are provided some assembly code generated by compiling the C functions using gcc compiler.

```
long f1(long a) {
  long b = 2*a;
  f2(&b, 3);
  return b;
}

void f2(long *p, long a) {
  if(a == 0)
    return;
  *p += a;
  f2(p, a-1);
  (*p)++;
}
```

```
f1:
  subq $24, %rsp
  addq %rdi, %rdi
  movq %rdi, 8(%rsp)
  movl $3, %esi
  leaq 8(%rsp), %rdi
  call f2
  movq 8(%rsp), %rax
  addq $24, %rsp
  ret

f2:
  testq %rsi, %rsi
  je .L5
  pushq %rbx
  movq %rdi, %rbx
  addq %rsi, (%rdi)
  subq $1, %rsi
  call f2
  addq $1, (%rbx)
  popq %rbx
.L5:
  rep ret
```

(a) [1 POINTS] What does `f1(6)` return?

21

(b) [9 POINTS] Assume that when `f1(6)` is called, just before the first instruction of `f1` is executed, the stack pointer `%rsp` has the value `0xFFFF1188`. Fill in this table to give the contents of registers immediately before the first instruction of `f2` is executed.

	`%rdi`	`%rsi`	`%rsp`
First call to `f2`	0xFFFF1178	0x3	0xFFFF1168
Second call to `f2`	0xFFFF1178	0x2	0xFFFF1158
Third call to `f2`	0xFFFF1178	0x1	0xFFFF1148
Fourth call to `f2`	0xFFFF1178	0x0	0xFFFF1138

(c) [1 POINTS] Does `f2` perform something to save and restore any caller saved registers? Yes/No

No

(d) [1 POINTS] Does `f2` perform something to save and restore any callee saved registers? Yes/No

Yes

(e) [1 POINTS] Does `f2` obey the x86-64/Linux calling conventions? Yes/No

Yes

Question 1B. Calling Functions in Assembly [16 POINTS]

In the following, you are provided some assembly code generated by compiling the C functions using gcc compiler.

```
long f1(long a) {
  long b = a+2;
  f2(3, &b);
  return b
}

void f2(long a, long *p) {
  if(a == 0)
    return;
  (*p)++;
  f2(a-1, p);
  *p += a;
}
```

```
f1:
  subq $24, %rsp
  addq $2, %rdi
  movq %rdi, 8(%rsp)
  leaq 8(%rsp), %rsi
  movl $3, %edi
  call f2
  movq 8(%rsp), %rax
  addq $24, %rsp
  ret

f2:
  testq %rdi, %rdi
  je .L5
  pushq %rbp
  pushq %rbx
  subq $8, %rsp
  movq %rsi, %rbx
  movq %rdi, %rbp
  addq $1, (%rsi)
  leaq -1(%rdi), %rdi
  call f2
  addq %rbp, (%rbx)
  addq $8, %rsp
  popq %rbx
  popq %rbp
.L5:
  rep ret
```

(a) [1 POINTS] What does `f1(4)` return?
15

(b) [12 POINTS] Assume that when `f1(4)` is called, just before the first instruction of `f1` is executed, the stack pointer `%rsp` has the value `0xFFFFCC80`. Fill in this table to give the contents of registers immediately before the first instruction of `f2` is executed.

	`%rdi`	`%rsi`	`%rsp`
First call to `f2`	0x3	0xFFFFCC70	0xFFFFCC60
Second call to `f2`	0x2	0xFFFFCC70	0xFFFFCC40
Third call to `f2`	0x1	0xFFFFCC70	0xFFFFCC20
Fourth call to `f2`	0x0	0xFFFFCC70	0xFFFFCC00

(c) [1 POINTS] Does `f2` perform something to save and restore any caller saved registers? Yes/No
No

(d) [1 POINTS] Does `f2` perform something to save and restore any callee saved registers? Yes/No
Yes

(e) [1 POINTS] Does `f2` obey the x86-64/Linux calling conventions? Yes/No
Yes

Question 2A. Data and Stack Frames [15 POINTS]

(a) [6 POINTS] Consider the following C program, in which `H` and `J` are constants expressed with `#define` directives, and the corresponding assembly code generated by the gcc compiler. By inspecting the assembly code, try to find the values of `H` and `J`.

```
int arr1[H][J];
int arr2[J][H];

void update_array(int x, int y) {
    arr2[y][x] = 3*arr1[x][y];
}
```

```
update_array:
  movslq %edi, %rdi
  movslq %esi, %rsi
  leaq (%rsi,%rsi,2), %rax
  addq %rdi, %rax
  leaq (%rdi,%rdi,8), %rdx
  addq %rdx, %rsi
  movl arr1(,%rsi,4), %edx
  leal (%rdx,%rdx,2), %edx
  movl %edx, arr2(,%rax,4)
  ret
```

`H` = 3

`J` = 9

(b) [9 POINTS] Suppose that you are developing an Instagram clone and implemented the following linked list data structure to store the uploaded photos. Determine the offset of each field and the total size (in bytes) of the structure given below, considering the alignment requirements of a 64-bit machine.

Structure	photo_id	user	likes	marked	date	next	text	Total

`typedef struct photo {`
`long photo_id;`
`char user[21];`
`int likes;`
`int marked;`
`int date;`
`photo* next;`
`char text[30];`
`} photo`
	0	**8**	**32**	**36**	**40**	**48**	**56**	**88**

What is the size (in bytes) of the structure if the fields in part (b) are rearranged to have minimum wasted space?
80

Question 2B. Data and Stack Frames [15 POINTS]

(c) [6 POINTS] Consider the following C program, in which `H` and `J` are constants expressed with `#define` directives, and the corresponding assembly code generated by the gcc compiler. By inspecting the assembly code, try to find the values of `H` and `J`.

```
int arr1[H][J];
int arr2[J][H];

void update_array(int x, int y) {
    arr2[y][x] = 6*arr1[x][y];
}
```

```
update_array:
  movslq %edi, %rdi
  movslq %esi, %rsi
  leaq (%rsi,%rsi,8), %rax
  addq %rdi, %rax
  leaq (%rdi,%rdi,2), %rdx
  addq %rdx, %rsi
  movl arr1(,%rsi,4), %edx
  leal (%rdx,%rdx,2), %edx
  addl %edx, %edx
  movl %edx, arr2(,%rax,4)
  ret
```

`H = 9`

`J = 3`

(d) [9 POINTS] Suppose that you are developing a Twitch TV clone and implemented the following linked list data structure to store the streamed videos. Determine the offset of each field and the total size (in bytes) of the structure given below, considering the alignment requirements of a 64-bit machine.

Structure	stream_id	user	views	live	date	next	title	Total

`typedef struct stream {`
`long stream_id;`
`char user[15];`
`int views;`
`short live;`
`int date;`
`stream* next;`
`char title[10];`
`} stream`
	0	**8**	**24**	**28**	**32**	**40**	**48**	**64**

What is the size (in bytes) of the structure if the fields in part (b) are rearranged to have minimum wasted space?
56

Question 3. Buffer Overflow [12 POINTS]

Richard Hendricks, the famous programmer who created the Pied Piper app wrote a program to test his new compression algorithm. The program reads a hexadecimal string from the standard input, converts it to an integer value, and then updates the value by incrementing it by one. This program uses an helper function `hex2int(char *buf,int n,char *result)`, which parses `2*n` hex ASCII digits into a n-byte integer value and stores the value in `*result`.

Hint: The program contains a small bug.

```
int read_and_update() {
   int n;
   char buf[20];
   fgets(buf, 20, stdin); // read input from terminal into buf
   hex2int(buf, 8, &n);   // parses hex ASCII digits to an 8-byte integer value into n
   n++;
   return n;
}

int main() {
   int x = read_and_update();
   printf("result is %d\n", x);
}
```

In the following, you are provided the assembly generated by compiling Richard Hendrick's program using gcc compiler code of is shown below. The parts marked with '…' are omitted for the sake of compactness.

```
000000000040063f <hex2int>:
  ...
  400647:       e8 da ff ff ff       callq  400626 <hex2byte>
  ...
  400652:       e8 cf ff ff ff       callq  400626 <hex2byte>
  ...
  40065e:      c3                    retq

00000000004006a6 <read_and_update>:
  4006a6:       48 83 ec 18          sub    $0x18,%rsp
  ...
  4006b1:       be 0a 00 00 00       mov    $0x14,%esi      //%esi: 2nd arg of fgets
  4006b6:       48 89 e7             mov    %rsp,%rdi       //%rdi: 1st arg of fgets
  4006b9:       e8 42 fe ff ff       callq  400500 <fgets@plt>
  4006be:       48 8d 54 24 0c       lea    0x14(%rsp),%rdx  //%rdx: 3rd arg of hex2int
  4006c3:       be 08 00 00 00       mov    $0x8,%esi       //%esi: 2nd arg. of hex2int
  4006c8:       48 89 e7             mov    %rsp,%rdi       //%rdi: 1st arg. of hex2int
  4006cb:       e8 8f ff ff ff       callq  40063f <hex2int>
  4006d0:       8b 44 24 0c          mov    0x14(%rsp),%eax
  4006d4:       83 c0 01             add    $0x1,%eax
  4006d7:       48 83 c4 18          add    $0x18,%rsp
  4006db:       c3                   retq
00000000004006dc <main>:
   ...
  4007b5:       b8 00 00 00 00       mov    $0x0,%eax
  4007ba:       e8 e7 fe ff ff       callq  4006a6 <read_and_update>
  4007bf:       89 44 24 0c          mov    %eax,0xc(%rsp)
   ...
  4007ef:       c3                   retq
```

(a) [4 POINTS] Does a buffer overflow happens in Richard's program?

4. The local variable `buf` in `read_and_update` might get overflown when `fgets` reads too many characters.
5. The local variable `x` in `main` might get overflown by `read_and_update`.
6. The local variable `n` in `read_and_update` might get overflown by `hex2int`
7. None of the above

(b) [4 POINTS] When the buffer overflow occurs, which one of the following addresses gets overwritten?

1. `0x00000000004006be`
2. `0x00000000004006d0`
3. `0x00000000004006cb`
4. `0x00000000004007bf`
5. None of the above

(c) [4 POINTS] Bertram Gilfoyle designs a malicious input string to exploit his boss' Richard's buffer overflow bug to hijack the control flow of the normal execution. When processing the malicious input, which one of the following is the last *normal* instruction executed by Richard's program before its control flow gets hijacked to execute code intended by Gilfoyle?

1. The `retq` instruction at address `0x00000000004006db`.
2. The `retq` instruction at address `0x00000000004007ef`.
3. The `retq` instruction at address `0x000000000040065e`.
4. None of the above.

Question 4. Locality [12 POINTS]

(a) [6 POINTS] Consider the following C functions which both calculate the transpose of a 2D matrix.

```
void calculateTranspose1(int N, int A[N][N]) {
        for (int i = 0; i < N; i++) {
           for (int j = 0; j < N; j++) {
              temp = A[i][j];
              A[i][j] = A[j][i];
              A[j][i] = temp;
           }
        }
}

void calculateTranspose2(int N, int A[N][N]) {
        for (int j = 0; j < N; j++) {
           for (int i = j+1; i < N; i++) {
              temp = A[i][j];
              A[i][j] = A[j][i];
              A[j][i] = temp;
           }
        }
}
```

The functions `calculateTranspose1` and `calculateTranspose2` accomplish the same task. Which one is more cache friendly?

1. `calculateTranspose1` is more cache friendly than `calculateTranspose2`
2. `calculateTranspose2` is more cache friendly than `calculateTranspose1`
3. **They perform similar in term of cache utilization.**

(a) [6 POINTS] Consider the following memory access requests given in a sequenced order.

Sequence A	Sequence B	Sequence C	Sequence D
0xFFFF0000	0xFFFF0000	0xFFFF0000	0xFFFF0000
0xFFFF0008	0xFFFF0004	0xFFFF0001	0xFFFF0000
0xFFFF0010	0xFFFF0008	0xFFFF0002	0xFFFF0000
0xFFFF0018	0xFFFF000C	0xFFFF0003	0xFFFF0000
0xFFFF0020	0xFFFF0010	0xFFFF0004	0xFFFF0000
0xFFFF0028	0xFFFF0014	0xFFFF0005	0xFFFF0008
0xFFFF0030	0xFFFF0018	0xFFFF0006	0xFFFF0008
0xFFFF0038	0xFFFF001C	0xFFFF0007	0xFFFF0008
0xFFFF0040	0xFFFF0020	0xFFFF0008	0xFFFF0008
0xFFFF0048	0xFFFF0024	0xFFFF0009	0xFFFF0008

Which of these sequences has the best spatial locality? **Sequence C**

Which of these sequences has the best temporal locality? **Sequence D**

Question 5A. Cache Memories [14 POINTS]

(a) [6 POINTS] Suppose that you are working on a system with the following specifications:

- The device is a 12-bit machine, i.e., the physical addresses are 12 bits wide.
- The memory is byte addressable, and memory accesses are to 1-byte words.
- The device contains a 2-way set associative cache (E = 2: Two lines per set)

Below table shows the current contents of the cache:

2-way Set Associate Cache (E = 2: Two lines per set)

Index	Valid	Tag	Byte 0	Byte1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Valid	Tag	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0	1	03	39	FC	FB	24	5F	5F	5F	42	0	1C	26	01	11	3F	31	4A	34	34
1	1	17	41	23	A3	67	34	23	1C	5F	1	0F	B3	1B	C5	B5	D3	E8	F5	B6
2	0	0F	44	43	28	92	93	94	4D	54	1	14	2E	8A	73	8D	9A	9B	9C	3A
3	1	0E	5D	24	8E	E8	8D	0F	3D	3E	1	07	54	3D	2A	8D	23	8A	8F	02
4	0	11	69	4A	77	4A	B1	21	23	65	1	12	72	9F	DE	4C	45	67	86	2C
5	0	20	74	4E	34	24	D1	44	55	39	1	0A	23	94	64	53	45	FF	75	D5
6	1	2C	8B	F3	56	93	34	48	6E	E4	0	1D	FF	FE	FD	FC	FB	FA	F9	48
7	0	0F	30	C3	B4	7E	4B	43	42	DD	0	0B	6D	1C	46	42	45	56	22	96

How many bits are used for the tag? **6 bits**

How many bits are used for the index? **3 bits**

For the access requests to the memory addresses given below, fill in the following table with the corresponding values for the offset *(O)*, index *(I)*, tag *(T)*, and whether a cache hit or miss occurred, and if it is a hit, what value will be returned? If miss write 'None' for the value.

Address to be read	Cache Offset (O)	Cache Index (I)	Cache Tag (T)	Cache Hit? (Y/N)	Cache Byte value
0x4A4	0x4	0x4	0x12	Y	0x45
0x3D7	0x7	0x2	0x0F	N	None

(b) [8 POINTS] Kinect is a motion sensing input device which is originally introduced by Microsoft in its Xbox console machines. By incorporating RGB cameras with infrared projectors and special sensors, it can capture depth images, where each pixel has an red, green, blue as well as a depth value. In the course project of your ELEC 317 Embedded Systems class, you are developing a software to effectively read and edit RGBD images of a 16×16 sensor grid. For that purpose, you have implemented the following struct and the program to initialize the array:

```
typedef struct rgbd_pixel {
        int r;
        int g;
        int b;
        int d;
} rgbd_pixel;

rgbd_pixel grid[16][16];
int i, j;
```

```
for (int i = 0; i < 16; i++) {
  for (int j = 0; j < 16; j++) {
        grid[i][j].r = 0;
        grid[i][j].g = 0;
        grid[i][j].b = 0;
        grid[i][j].d = 0;
    }
}
```

Suppose that the hardware you are working on has a 2048-byte direct-mapped data cache with 32-byte blocks, which is initially empty, and the size of an `int` is 4 bytes, and the array `grid` begins at memory address `0x0`, and the variables `i` and `j` are stored in registers. What is the hit ratio when the initialization code given above is executed?

7/8 The size of grid is 16x16x4x4 = 4096 (twice of the size of the cache), and each cache block can store 2 structs. Hence, after each grid[i][j].r miss there will be a hit for that and the following struct.

Question 5B. Cache Memories [14 POINTS]

(a) [6 POINTS] Suppose that you are working on a system with the following specifications:

- The device is a 12-bit machine, i.e., the physical addresses are 12 bits wide.
- The memory is byte addressable, and memory accesses are to 1-byte words.
- The device contains a 2-way set associative cache (E = 2: Two lines per set)

Below table shows the current contents of the cache:

2-way Set Associate Cache (E = 2: Two lines per set)

Index	Valid	Tag	Byte 0	Byte1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Valid	Tag	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
0	1	03	39	FC	FB	24	5F	5F	5F	42	0	1C	26	01	11	3F	31	4A	34	34
1	1	17	41	23	A3	67	34	23	1C	5F	1	0F	B3	1B	C5	B5	D3	E8	F5	B6
2	0	0F	44	43	28	92	93	94	4D	54	1	14	2E	8A	73	8D	9A	9B	9C	3A
3	1	0E	5D	24	8E	E8	8D	0F	3D	3E	1	07	54	3D	2A	8D	23	8A	8F	02
4	0	11	69	4A	77	4A	B1	21	23	65	1	12	72	9F	DE	4C	45	67	86	2C
5	0	20	74	4E	34	24	D1	44	55	39	1	0A	23	94	64	53	45	FF	75	D5
6	1	2C	8B	F3	56	93	34	48	6E	E4	0	1D	FF	FE	FD	FC	FB	FA	F9	48
7	0	0F	30	C3	B4	7E	4B	43	42	DD	0	0B	6D	1C	46	42	45	56	22	96

How many bits are used for the tag? **6 bits**

How many bits are used for the index? **3 bits**

For the access requests to the memory addresses given below, fill in the following table with the corresponding values for the offset *(O)*, index *(I)*, tag *(T)*, and whether a cache hit or miss occurred, and if it is a hit, what value will be returned? If miss write 'None' for the value.

Address to be read	Cache Offset (O)	Cache Index (I)	Cache Tag (T)	Cache Hit? (Y/N)	Cache Byte value
0x515	0x5	0x2	0x14	Y	0x9B
0x467	0x7	0x4	0x11	N	None

(b) [8 POINTS] In your COMP 341 Introduction to Artificial Intelligence class, you are developing a chess program as your course project. For this project, you decided to implement a simple GUI using Unicode symbols (e.g. ♘). Considering that the chessboard is composed of 8×8 squares, you have implemented the following struct to store the pieces in a square and whether that square is visited before or not, and a program to initialize the 2D array:

```
typedef struct square {
        int visited;
        int piece;
} square;

square chessboard[8][8];
int i, j;
```

```
for (int i = 0; i < 8; i++) {
  for (int j = 0; j < 8; j++) {
        board[i][j].visited = 0;
        board[i][j].piece = 0;
    }
}
```

Suppose that the hardware you are working on has a 256-byte direct-mapped data cache with 16-byte blocks, which is initially empty, and the size of an `int` is 4 bytes, and the array `board` begins at memory address `0x0`, and the variables `i` and `j` are stored in registers. What is the hit ratio when the initialization code given above is executed?

```
3/4     The size of grid is 8x8x2x4 = 512 (twice of the size of the cache), and each
        each cache block can store 2 structs. Hence, after each board[i][j].visited
        miss there will be a hit for that and the next struct.
```

Question 6A. Assembly and Linking [18 POINTS]

(a) [15 POINTS] Fill in the missing parts of the C functions based on the corresponding Assembly code generated by gcc.

mylib.c

```
extern int a;

int myfun1(int x) {
   return ((x != a) && (32*x < 64));
}
```

mylib.s

```
myfun1:
  cmpl %edi, a(%rip)
  je .L3
  sall $5, %edi
  cmpl $63, %edi
  jle .L4
  movl $0, %eax
  ret
.L3:
  movl $0, %eax
  ret
.L4:
  movl $1, %eax
  ret
```

myfile.c

```
int a;

int myfun1(int);

typedef struct mystruct {
  struct mystruct *a;
  int b;
} mystruct;

int myfun2(mystruct *s, int b) {
  int y = 0;

  while (s != NULL) {
    if (myfun1(s->b)) {
      y = s->b + b*15;
    }
    s = s->a;
  }
  return y;
}
```

myfile.s

```
myfun2:
  ...
  testq %rdi, %rdi
  je .L5
  movq %rdi, %rbx
  movl %esi, %ebp
  sall $4, %ebp
  subl %esi, %ebp
  movl $0, %r12d
.L4:
  movl 8(%rbx), %edi
  call myfun1
  testl %eax, %eax
  je .L3
  movl %ebp, %r12d
  addl 8(%rbx), %r12d
.L3:
  movq (%rbx), %rbx
  testq %rbx, %rbx
  jne .L4
  jmp .L2
.L5:
  movl $0, %r12d
.L2:
  movl %r12d, %eax
  ...
  ret
```

Fill in the following table to name the strong, weak and external symbols defined in the C files. If there is no such definition just write 'None'. If there are more than one, separate the names with commas.

The strong symbols defined in `mylib.c`	The weak symbols defined in `mylib.c`	The external symbols defined in `mylib.c`	The strong symbols defined in `myfile.c`	The weak symbols defined in `myfile.c`	The external symbols defined in `myfile.c`
myfun1	None	a	myfun2	a	myfun1

(b) [3 POINTS] Consider the following two C files named `myfile1.c` and `myfile2.c`.

myfile1.c

```
float x;

void dec();

int main(void) {
  printf("x = %.3f\n", x);
}
```

myfile2.c

```
int x = 4;

void dec() {
  x--;
}
```

What happens if we compile these programs using `gcc -o prog myfile1.c myfile2.c` and then run the program?

1. Compile-time error! The function `dec` has been defined twice.
2. Compile-time error! The global variable `x` has been defined twice.
3. Run-time error! The program produces a segmentation fault.
4. The program will output `x = 3.000`.
5. **The program will outputs some number, but different from `x = 3.000`.**

Question 6B. Assembly and Linking [18 POINTS]

(a) [15 POINTS] Fill in the missing parts of the C functions based on the corresponding Assembly code generated by gcc.

mylib.c

```
extern int a;

int myfun1(int x) {
   return ((x <= a) || (32*x == 32));
}
```

mylib.s

```
myfun1:
  cmpl %edi, a(%rip)
  jge .L3
  sall $5, %edi
  cmpl $32, %edi
  jne .L4
  movl $1, %eax
  ret
.L3:
  movl $1, %eax
  ret
.L4:
  movl $0, %eax
  ret
```

myfile.c

```
int a;

int myfun1(int);

typedef struct mystruct {
  struct mystruct *next;
  int a;
} mystruct;

int myfun2(mystruct *s, int b) {
  int y = 0;
  while (s->next) {
    s = s->next;
    if (myfun1(s->a)) {
      y = s->a + (b*15);
    }
  }
  return y;
}
```

myfile.s

```
myfun2:
  ...
  movq %rdi, %rbx
  movl %esi, %ebp
  sall $4, %ebp
  subl %esi, %ebp
  movl $0, %r12d
  jmp .L2
.L4:
  movl 8(%rbx), %edi
  call myfun1
  testl %eax, %eax
  je .L2
  movl %ebp, %r12d
  addl 8(%rbx), %r12d
.L2:
  movq (%rbx), %rbx
  testq %rbx, %rbx
  jne .L4
  movl %r12d, %eax
  ...
  ret
```

Fill in the following table to name the strong, weak and external symbols defined in the C files. If there is no such definition just write 'None'. If there are more than one, separate the names with commas.

The strong symbols defined in mylib.c	The weak symbols defined in mylib.c	The external symbols defined in mylib.c	The strong symbols defined in myfile.c	The weak symbols defined in myfile.c	The external symbols defined in myfile.c
myfun1	None	a	myfun2	a	myfun1

(b) [3 POINTS] Consider the following two C files named `myfile1.c` and `myfile2.c`.

myfile1.c
```
float x = 4.0;

void inc();

int main(void) {
  printf("x = %.3f\n", x);
}
```

myfile2.c
```
int x = 4;

void inc() {
  x++;
}
```

What happens if we compile these programs using `gcc -o prog myfile1.c myfile2.c` and then run the program?

1. Compile-time error! The function `inc` has been defined twice.
2. Compile-time error! The global variable x has been defined twice.
3. Run-time error! The program produces a segmentation fault.
4. The program will output `x = 5.000`.
5. The program will outputs some number, but different from `x = 5.000`.

Question 7. Memory Allocators [10 POINTS]

Consider the following state of the heap with *an implicit free-list allocator with coalescing*. The header is of 8 bytes and the allocated chunks of memory should be multiples of 8 bytes, and a first-fit policy is employed. For the sake of simplicity, the memory addresses are given in decimal numbers.

Address:	72	80	88	96	104	112	120	128	136	144	152	160	168	176
Memory:	Free, 8		Used, 16			Free, 24				Used, 16			Used, 8	

Address:	184	192	200	208	216	224	232	240	248	256	264	272	280	288
Memory:	Free, 16			Used, 8		Free, 16			Used, 16			Used, 16		

(a) [2 POINTS] Suppose that the next call to the heap allocator is `malloc(12)`. Specify the address that the malloc function returns?

120

(b) [2 POINTS] What if the best-fit approach were used instead for the request in part (a)?

192

(c) [2 POINTS] What is the amount of internal fragmentation for the malloc request given in part (a) regardless of the placement policy used in allocation?

4 bytes

(d) [2 POINTS] Best-fit placement policy always provides a better utilization and throughput performance as compared to first-fit policy. True/ False

False. It only provides better utilization

(e) [2 POINTS]) Suppose that the next call to the heap allocator is `free(208)`. After this operation, can `malloc(32)` request be satisfied? Yes/No

Yes. Because of coalescing, there is enough free space.