CS 210 PR Problem Set Part B

Jeong Yong Yang

TOTAL POINTS

22 / 22

QUESTION 1

Problem 4 12 pts

1.1 400626 **1 / 1**

✓ **- 0 pts** Correct

1.2 40062b **1 / 1**

✓ **- 0 pts** Correct

1.3 40062d **2 / 2**

✓ **- 0 pts** Correct

1.4 400630 **2 / 2**

✓ **- 0 pts** Correct

1.5 400634 **2 / 2**

✓ **- 0 pts** Correct

1.6 400637 **1 / 1**

✓ **- 0 pts** Correct

1.7 What purpose does the mystery function serve eg. What is it doing? **3 / 3**

✓ **- 0 pts** Correct

QUESTION 2

Problem 6 10 pts

2.1 blank for x **3 / 3**

✓ **- 0 pts** Correct

2.2 blank for y **3 / 3**

✓ **- 0 pts** Correct

2.3 m = blank **4 / 4**

✓ **- 0 pts** Correct

Given the above code and the following objdump output

```
<mystery>:
 000000000040061f <mystery>:
  40061f:       48 8b 15 1a 0a 20 00     mov    0x200a1a(%rip),%rdx   # 601040 <Emp_list>
  400626:       b8 00 00 00 00           mov    $0x0,%eax
  40062b:       eb 07                    jmp    400634 <mystery+0x15>
  40062d:       03 42 58                 add    0x58(%rdx),%eax
  400630:       48 8b 52 60              mov    0x60(%rdx),%rdx
  400634:       48 85 d2                 test   %rdx,%rdx
  400637:       75 f4                    jne    40062d <mystery+0xe>
  400639:       c3                       retq
```

Assuming `&Emp list` is `0x601040` fill in the following table. Your explanations should not just be a restatement of the assembly code. Rather the explanation sould be interms of the what the assembly is doing in context of the above 'C' code. Here are two examples of the kind of explanations we are looking for: 1) "load rdx with the employee id" and 2) "test if the list is empty". Note: Use x86 64 alignment rules thus pointers are 8 bytes in size and 8 bytes aligned.

Address	Explanation
40061f	initialize rdx to value of Emp list
400626	Initialize the return value (eax) to 0
40062b	Jump to the address 400634, where the test of whether to run the loop or not happens
40062d	Add the current employee's salary to the return value
400630	Move rdx to point the next employee in the list
400634	Test if we are at the end of the list (there are no more values to compare)
400637	If we are at the end of the list, go to the next line. If we are not at the end of the list, jump to the top of the loop
400639	return

What purpose does the mystery function server eg. what is it doing?

The purpose of the mystery function is to add all the salary of the employees in the list.

1.1 400626 **1 / 1**

✓ **- 0 pts** Correct

Given the above code and the following objdump output

```
<mystery>:
 000000000040061f <mystery>:
  40061f:       48 8b 15 1a 0a 20 00     mov    0x200a1a(%rip),%rdx  # 601040 <Emp_list>
  400626:       b8 00 00 00 00           mov    $0x0,%eax
  40062b:       eb 07                    jmp    400634 <mystery+0x15>
  40062d:       03 42 58                 add    0x58(%rdx),%eax
  400630:       48 8b 52 60              mov    0x60(%rdx),%rdx
  400634:       48 85 d2                 test   %rdx,%rdx
  400637:       75 f4                    jne    40062d <mystery+0xe>
  400639:       c3                       retq
```

Assuming `&Emp list` is `0x601040` fill in the following table. Your explanations should not just be a restatement of the assembly code. Rather the explanation sould be interms of the what the assembly is doing in context of the above 'C' code. Here are two examples of the kind of explanations we are looking for: 1) "load rdx with the employee id" and 2) "test if the list is empty". Note: Use x86 64 alignment rules thus pointers are 8 bytes in size and 8 bytes aligned.

Address	Explanation
40061f	initialize rdx to value of Emp list
400626	Initialize the return value (eax) to 0
40062b	Jump to the address 400634, where the test of whether to run the loop or not happens
40062d	Add the current employee's salary to the return value
400630	Move rdx to point the next employee in the list
400634	Test if we are at the end of the list (there are no more values to compare)
400637	If we are at the end of the list, go to the next line. If we are not at the end of the list, jump to the top of the loop
400639	return

What purpose does the mystery function server eg. what is it doing?

The purpose of the mystery function is to add all the salary of the employees in the list.

1.2 40062b **1 / 1**

✓ **- 0 pts** Correct

Given the above code and the following objdump output

```
<mystery>:
 000000000040061f <mystery>:
  40061f:       48 8b 15 1a 0a 20 00     mov    0x200a1a(%rip),%rdx  # 601040 <Emp_list>
  400626:       b8 00 00 00 00           mov    $0x0,%eax
  40062b:       eb 07                    jmp    400634 <mystery+0x15>
  40062d:       03 42 58                 add    0x58(%rdx),%eax
  400630:       48 8b 52 60              mov    0x60(%rdx),%rdx
  400634:       48 85 d2                 test   %rdx,%rdx
  400637:       75 f4                    jne    40062d <mystery+0xe>
  400639:       c3                       retq
```

Assuming `&Emp list` is `0x601040` fill in the following table. Your explanations should not just be a restatement of the assembly code. Rather the explanation sould be interms of the what the assembly is doing in context of the above 'C' code. Here are two examples of the kind of explanations we are looking for: 1) "load rdx with the employee id" and 2) "test if the list is empty". Note: Use x86 64 alignment rules thus pointers are 8 bytes in size and 8 bytes aligned.

Address	Explanation
40061f	initialize rdx to value of Emp list
400626	Initialize the return value (eax) to 0
40062b	Jump to the address 400634, where the test of whether to run the loop or not happens
40062d	Add the current employee's salary to the return value
400630	Move rdx to point the next employee in the list
400634	Test if we are at the end of the list (there are no more values to compare)
400637	If we are at the end of the list, go to the next line. If we are not at the end of the list, jump to the top of the loop
400639	return

What purpose does the mystery function server eg. what is it doing?

The purpose of the mystery function is to add all the salary of the employees in the list.

1.3 40062d **2 / 2**

✓ **- 0 pts** Correct

Given the above code and the following objdump output

```
<mystery>:
 000000000040061f <mystery>:
  40061f:       48 8b 15 1a 0a 20 00     mov    0x200a1a(%rip),%rdx   # 601040 <Emp_list>
  400626:       b8 00 00 00 00           mov    $0x0,%eax
  40062b:       eb 07                    jmp    400634 <mystery+0x15>
  40062d:       03 42 58                 add    0x58(%rdx),%eax
  400630:       48 8b 52 60              mov    0x60(%rdx),%rdx
  400634:       48 85 d2                 test   %rdx,%rdx
  400637:       75 f4                    jne    40062d <mystery+0xe>
  400639:       c3                       retq
```

Assuming `&Emp list` is `0x601040` fill in the following table. Your explanations should not just be a restatement of the assembly code. Rather the explanation sould be interms of the what the assembly is doing in context of the above 'C' code. Here are two examples of the kind of explanations we are looking for: 1) "load rdx with the employee id" and 2) "test if the list is empty". Note: Use x86 64 alignment rules thus pointers are 8 bytes in size and 8 bytes aligned.

Address	Explanation
40061f	initialize rdx to value of Emp_list
400626	Initialize the return value (eax) to 0
40062b	Jump to the address 400634, where the test of whether to run the loop or not happens
40062d	Add the current employee's salary to the return value
400630	Move rdx to point the next employee in the list
400634	Test if we are at the end of the list (there are no more values to compare)
400637	If we are at the end of the list, go to the next line. If we are not at the end of the list, jump to the top of the loop
400639	return

What purpose does the mystery function server eg. what is it doing?

The purpose of the mystery function is to add all the salary of the employees in the list.

1.4 400630 **2 / 2**

✓ **- 0 pts** Correct

Given the above code and the following objdump output

```
<mystery>:
 000000000040061f <mystery>:
  40061f:       48 8b 15 1a 0a 20 00     mov    0x200a1a(%rip),%rdx  # 601040 <Emp_list>
  400626:       b8 00 00 00 00           mov    $0x0,%eax
  40062b:       eb 07                    jmp    400634 <mystery+0x15>
  40062d:       03 42 58                 add    0x58(%rdx),%eax
  400630:       48 8b 52 60              mov    0x60(%rdx),%rdx
  400634:       48 85 d2                 test   %rdx,%rdx
  400637:       75 f4                    jne    40062d <mystery+0xe>
  400639:       c3                       retq
```

Assuming `&Emp list` is `0x601040` fill in the following table. Your explanations should not just be a restatement of the assembly code. Rather the explanation sould be interms of the what the assembly is doing in context of the above 'C' code. Here are two examples of the kind of explanations we are looking for: 1) "load rdx with the employee id" and 2) "test if the list is empty". Note: Use x86 64 alignment rules thus pointers are 8 bytes in size and 8 bytes aligned.

Address	Explanation
40061f	initialize rdx to value of Emp list
400626	Initialize the return value (eax) to 0
40062b	Jump to the address 400634, where the test of whether to run the loop or not happens
40062d	Add the current employee's salary to the return value
400630	Move rdx to point the next employee in the list
400634	Test if we are at the end of the list (there are no more values to compare)
400637	If we are at the end of the list, go to the next line. If we are not at the end of the list, jump to the top of the loop
400639	return

What purpose does the mystery function server eg. what is it doing?

The purpose of the mystery function is to add all the salary of the employees in the list.

1.5 400634 **2 / 2**

✓ **- 0 pts** Correct

Given the above code and the following objdump output

```
<mystery>:
 000000000040061f <mystery>:
  40061f:       48 8b 15 1a 0a 20 00     mov    0x200a1a(%rip),%rdx   # 601040 <Emp_list>
  400626:       b8 00 00 00 00           mov    $0x0,%eax
  40062b:       eb 07                    jmp    400634 <mystery+0x15>
  40062d:       03 42 58                 add    0x58(%rdx),%eax
  400630:       48 8b 52 60              mov    0x60(%rdx),%rdx
  400634:       48 85 d2                 test   %rdx,%rdx
  400637:       75 f4                    jne    40062d <mystery+0xe>
  400639:       c3                       retq
```

Assuming `&Emp list` is `0x601040` fill in the following table. Your explanations should not just be a restatement of the assembly code. Rather the explanation sould be interms of the what the assembly is doing in context of the above 'C' code. Here are two examples of the kind of explanations we are looking for: 1) "load rdx with the employee id" and 2) "test if the list is empty". Note: Use x86 64 alignment rules thus pointers are 8 bytes in size and 8 bytes aligned.

Address	Explanation
40061f	initialize rdx to value of Emp_list
400626	Initialize the return value (eax) to 0
40062b	Jump to the address 400634, where the test of whether to run the loop or not happens
40062d	Add the current employee's salary to the return value
400630	Move rdx to point the next employee in the list
400634	Test if we are at the end of the list (there are no more values to compare)
400637	If we are at the end of the list, go to the next line. If we are not at the end of the list, jump to the top of the loop
400639	return

What purpose does the mystery function server eg. what is it doing?

The purpose of the mystery function is to add all the salary of the employees in the list.

1.6 400637 **1 / 1**

✓ **- 0 pts** Correct

Given the above code and the following objdump output

```
<mystery>:
 000000000040061f <mystery>:
  40061f:       48 8b 15 1a 0a 20 00     mov    0x200a1a(%rip),%rdx  # 601040 <Emp_list>
  400626:       b8 00 00 00 00           mov    $0x0,%eax
  40062b:       eb 07                    jmp    400634 <mystery+0x15>
  40062d:       03 42 58                 add    0x58(%rdx),%eax
  400630:       48 8b 52 60              mov    0x60(%rdx),%rdx
  400634:       48 85 d2                 test   %rdx,%rdx
  400637:       75 f4                    jne    40062d <mystery+0xe>
  400639:       c3                       retq
```

Assuming `&Emp list` is `0x601040` fill in the following table. Your explanations should not just be a restatement of the assembly code. Rather the explanation sould be interms of the what the assembly is doing in context of the above 'C' code. Here are two examples of the kind of explanations we are looking for: 1) "load rdx with the employee id" and 2) "test if the list is empty". Note: Use x86 64 alignment rules thus pointers are 8 bytes in size and 8 bytes aligned.

Address	Explanation
40061f	initialize rdx to value of Emp list
400626	Initialize the return value (eax) to 0
40062b	Jump to the address 400634, where the test of whether to run the loop or not happens
40062d	Add the current employee's salary to the return value
400630	Move rdx to point the next employee in the list
400634	Test if we are at the end of the list (there are no more values to compare)
400637	If we are at the end of the list, go to the next line. If we are not at the end of the list, jump to the top of the loop
400639	return

What purpose does the mystery function server eg. what is it doing?

The purpose of the mystery function is to add all the salary of the employees in the list.

1.7 What purpose does the mystery function serve eg. What is it doing? **3 / 3**

✓ **- 0 pts** Correct

Problem 6: 10 Points

Consider the following incomplete definition of a C struct along with the incomplete code for a function `func` given below.

```
typedef struct node {

  double x;

  unsigned short y;

  struct node *next;

  struct node *prev;

} node_t;
```

```
node_t n;

void func() {

  node_t *m;

  m = n.next -> prev

  m->y /= 16;

  return;
}
```

When this C code was compiled on an IA-64 machine running Linux, the following assembly code was generated for function `func`.

```
func:
  movq n+16(%rip),%rax
  movq 24(%rax),%rax
  shrw $0x4,8(%rax)
  retq
```

Given these code fragments, fill in the blanks in the C code given above. Note that there is a unique answer.

The types must be chosen from the following table, assuming the sizes and alignment given.

Type	Size (bytes)	Alignment (bytes)
char	1	1
short	2	2
unsigned short	2	2
int	4	4
unsigned int	4	4
double	8	8

2.1 blank for x 3 / 3

✓ - 0 pts Correct

Problem 6: 10 Points

Consider the following incomplete definition of a C struct along with the incomplete code for a function `func` given below.

```
typedef struct node {

  double x;

  unsigned short y;

  struct node *next;

  struct node *prev;

} node_t;
```

```
node_t n;

void func() {

  node_t *m;

  m = n.next -> prev

  m->y /= 16;

  return;
}
```

When this C code was compiled on an IA-64 machine running Linux, the following assembly code was generated for function `func`.

```
func:
  movq n+16(%rip),%rax
  movq 24(%rax),%rax
  shrw $0x4,8(%rax)
  retq
```

Given these code fragments, fill in the blanks in the C code given above. Note that there is a unique answer.

The types must be chosen from the following table, assuming the sizes and alignment given.

Type	Size (bytes)	Alignment (bytes)
char	1	1
short	2	2
unsigned short	2	2
int	4	4
unsigned int	4	4
double	8	8

2.2 blank for y **3 / 3**

✓ **- 0 pts** Correct

Problem 6: 10 Points

Consider the following incomplete definition of a C struct along with the incomplete code for a function `func` given below.

```
typedef struct node {

  double x;

  unsigned short y;

  struct node *next;

  struct node *prev;

} node_t;
```

```
node_t n;

void func() {

  node_t *m;

  m = n.next -> prev

  m->y /= 16;

  return;
}
```

When this C code was compiled on an IA-64 machine running Linux, the following assembly code was generated for function `func`.

```
func:
  movq n+16(%rip),%rax
  movq 24(%rax),%rax
  shrw $0x4,8(%rax)
  retq
```

Given these code fragments, fill in the blanks in the C code given above. Note that there is a unique answer.

The types must be chosen from the following table, assuming the sizes and alignment given.

Type	Size (bytes)	Alignment (bytes)
char	1	1
short	2	2
unsigned short	2	2
int	4	4
unsigned int	4	4
double	8	8

2.3 m = blank **4 / 4**

✓ **- 0 pts** Correct