Spring Semester 2024

KAIST EE209

Programming Structures for Electrical Engineering

# Midterm Exam

Date: 2024.04.15

Time: 13:00 ~ 14:10

Student ID:

Name:

The exam is closed book and notes. Read the questions carefully and focus your answers on what has been asked. You are allowed to ask the instructor/TAs for help only in understanding the questions, in case you find them not completely clear. Be concise and precise in your answers and state clearly any assumption you may have made. All your answers must be included in the attached sheets. You have 70 minutes to complete your exam. Be wise in managing your time.
Please do not fill in the "Score" fields below. Self-grading is not allowed. Good luck!

| | |
|---|---|
| 1 | /10 |
| 2 | /10 |
| 3 | /15 |
| 4 | /10 |
| 5 | /8 |
| 6 | /18 |
| 7 | /10 |
| 8 | /10 |
| 9 | /9 |
| Total | /100 |

Student ID:                    Name:

1. Please answer the following questions. (10 pts)

   **For Question 1, assume all programs in this exam run on 64-bit Linux (x86-64). That is, assume that the code runs on eelabg1.kaist.ac.kr**

   a. What is the output of this code? (1 pt)

```
unsigned int a = 0xffffffff;
printf("%d\n", a + 8);
```

      i. 7
      ii. 8
      iii. 9
      iv. None of above (due to undefined behaviors)

   Answer: i

   b. What is the output of this code? (1 pt)

```
int a, b;
printf("%d\n", a=b=3);
```

      i. 3
      ii. 1
      iii. 0
      iv. None of above (due to undefined behaviors)
      v. compile error

   Answer: i

Student ID:                    Name:

c. What's the output of this code snippet (output of `f1(a, b, c)`)? (`%zu` prints the return value of the `sizeof` function.) (6 pts)

```
void f(int a[5], int b[], int *c) {
     printf("1: %zu 2:%zu 3:%zu 4:%zu 5:%zu 6:%zu \n",
                sizeof(a), sizeof(b), sizeof(c),
                sizeof(*a), sizeof(*b), sizeof(*c));
}

int A[5] = {1, 2, 3, 4, 5};
int B[3] = {1, 2, 3};
int *C = {1, 2, 3, 4};
f(A, B, C);
```

Output: (1 pt for each)
1: _____
2: _____
3: _____
4: _____
5: _____
6: _____

Answer: 1~3: 8, 4~6: 4

d. What's the output of this code snippet? (2 pts)

```
int i, j, r1, r2, r3;
i = 36;
j = 144;
r1 = i ^ j;
r2 = r1 >> 2;
r3 = r1 & r2;
printf("r1=%d, r3=%d\n", r1, r3);
```

Output: (1 pt for each)
r1: _______
r3: _______

Answer: r1=180, r3=36

Student ID:                    Name:

2. Please implement the following functions. (10pts)

   a. Implement `strncmp(const char* p1, const char* p2, int n)`. This function does the same job as `strcmp()` except that it compares only the first `n` characters of the two input strings. This function compares the first `n` characters of the input strings, `p1` and `p2`, and returns a negative(/positive) integer if `p1` comes earlier(/later) than `p2` or 0 if they are the same. (For the purpose of this problem, **assume that the return value of** `strcmp()` **or** `strncmp()` **is one of -1, 0, and 1.**) For full credit, your code should be both correct and efficient. (No need to handle the input errors such as NULL pointers for `p1`, `p2` or a non-positive `n`. ) (5 pts)

```
int strncmp(const char *p1, const char *p2, size_t n)
{
      unsigned char c1, c2;
      /* fill the code below*/









        return 0;
}
```

Answer:

```
while (n) {
      c1 = *p1++;
             c2 = *p2++;
             if (c1 != c2)
                     return c1 < c2 ? -1 : 1;
             if (!c1)
                     break;
             count--;
  }
```

**Grading Criteria.**

Show correct behaviors of the function: 5pt
1 pt deduction for all the minor syntax errors (-1pt at max)
    -typo, missing semicolon, bracket, etc
1 pt deduction each for each wrong behavior (-4pt at max)
    -opposite return values
    -wrong return values for corner cases
    -wrong comparison of characters
    -wrong iteration
    -etc
Otherwise (Answer is incomplete or not functional): 0pt

Student ID: Name:

b. Implement `strncpy(char *dest, const char *src, size_t n);` which copies `n` bytes of src to dest. It is similar to `strcpy()`, except that at most `n` bytes of src are copied. Warning: If there is no NULL byte among the first `n` bytes of `src`, the string placed in `dest` will not be NULL-terminated. If the length of `src` is less than `n`, `strncpy()` writes additional NULL bytes to `dest` to ensure that a total of `n` bytes are written. Write the body of the function. Of course, you should not call any C runtime library function inside the body. For full credit, your code should be both correct and efficient. (No need to handle the input errors such as NULL pointers for `dest`, `src` or a non-positive `n`. ) (5 pts)

```
char* strncpy(char* dest, const char *src, size_t n)
{
    /* fill the code below*/













    return dest;
}
```

Answer:

```
char *p = dest;
size_t i;

while (n && *src) {
    *p++ = *src++;
    n--;
 }

for (i = 0; i < n; i++)
         p++ = 0;
```

**Grading Criteria.**

Show correct behaviors of the function: 5pt

1 pt deduction for all the minor syntax errors (-1pt at max)

-typo, missing semicolon, bracket, etc

1 pt deduction each for each wrong behavior (-4pt at max)

-making dest NULL-terminated even if there is no NULL byte among the first n bytes of src

-missing additional NULL bytes for dest to ensure that a total of n bytes are written

-wrong copy of bytes

-wrong iteration

-wrong inference to src or dest

-wrong results at corner cases

-etc

Otherwise (Answer is incomplete or not functional): 0pt

Student ID: Name:

3. Please refer to the code below. (15pts)

```
void swap(int* a, int* b){ int t = *a; *a = *b; *b = t; }
/* Partition takes array, sets the pivot element to arr[high] and
rearranges an array such that items less than or equal to the pivot
stays at the left of the array. It returns the pivot position */

int partition (int arr[], int low, int high)
{
   int pivot = arr[high];
   int i = (low - 1);
   for (int j = low; j <= high- 1; j++) {
      if (arr[j] <= pivot) {
         i++;
         swap(&arr[i], &arr[j]); // smaller than pivot
      }
   }
   swap(&arr[i + 1], &arr[high]); // move pivot to arr[i+1]
   return (i + 1); // pivot location
}
```

a. Fill in the blank to complete the `myquicksort` function that performs quicksort. use the function(s) given above. Do nothing when the `low` is greater than or equal to `high.` (5 pts)

```
void myquicksort (int arr[], int low, int high){

      Answer :
            if (low < high) {
                 int pi = partition(arr, low, high);

                 myquicksort(arr, low, pi - 1); // Before pi
                 myquicksort(arr, pi + 1, high); // After pi
               }

syntax error: -1pt

}
```

Student ID: Name:

b. Consider the following integer array `arr`. You sort the array arr with the `myquicksort` you have written in the question above. Show the contents of array `arr[]` each time after the `partition` is called. Assume that `low` is 0 and `high` is 9. (5 pts)

```
int arr[] = {90, 20, 37, 55, 41, 67, 15, 4, 18, 29};
```

Answer :
20 15 4 18 29 67 90 37 55 41
15 4 18 20 29 67 90 37 55 41
4 15 18 20 29 67 90 37 55 41
4 15 18 20 29 37 41 67 55 90
4 15 18 20 29 37 41 67 55 90
4 15 18 20 29 37 41 55 67 90

Student ID: Name:

c. Consider the following integer array `arr`. You sort the array `arr[]` with the `myquicksort` you have written in the question above. Show the contents of array `arr[]` each time after the `partition` is called. Assume that `low` is 0 and `high` is 9. (5 pts)

```
int arr[] = {4, 15, 18, 20, 29, 37, 41, 55, 67, 90}
```

Answer :

4 15 18 20 29 37 41 55 67 90
4 15 18 20 29 37 41 55 67 90
4 15 18 20 29 37 41 55 67 90
4 15 18 20 29 37 41 55 67 90
4 15 18 20 29 37 41 55 67 90
4 15 18 20 29 37 41 55 67 90
4 15 18 20 29 37 41 55 67 90
4 15 18 20 29 37 41 55 67 90
4 15 18 20 29 37 41 55 67 90

Student ID: Name:

4. For the following code, similar to what was presented during the lecture. What value of `i` will be printed by Line A in this code? (10 pts.)

```
#include <stdio.h>
int i;

void print_one_row(void)
{
  for (i = 1; i <= 10; i++)
      printf("*");
}

void print_all_rows(void)
{
  for (i = 1; i <= 10; i++) {
      printf("1. i = %d\n",i);
      print_one_row();
      printf("\n");
  }
}

int main() {
      print_all_rows();
      printf("i = %d\n",i); // -------- Line A
}
```

ANSWER : i = 12 (partial 2pts if i = 11 is provided)

Student ID: Name:

5. Please show the output of the program below. (8pts, 1pt each)

```
#include <stdio.h>
int var = 0;
int f1(int var) {
      var +=1 ;
      return var ;
}
int f2() {
      var +=3 ;
      return var ;
}
int f3() {
      int var = 0 ;
      var += 100 ;
      return var ;
}
int f4() {
      static int var = 0 ;
      var += 17 ;
      if ( var <= 20 ) {
            return (var + 3);
      }
      return (var + 6);
}
int main() {
      printf("f1 (1): %d\n", f1(1));
      printf("f1 (2): %d\n", f1(1));
      printf("f2 (1): %d\n", f2());
      printf("f2 (2): %d\n", f2());
      printf("f3 (1): %d\n", f3());
      printf("f3 (2): %d\n", f3());
      printf("f4 (1): %d\n", f4());
      printf("f4 (2): %d\n", f4());
}
```

Answer : 1pt deduction for wrong format

```
f1 (1): 2
f1 (2): 2
f2 (1): 3
f2 (2): 6
f3 (1): 100
f3 (2): 100
f4 (1): 20
f4 (2): 40
```

Student ID:                    Name:

6. Answer the following questions (18pts).
   a. When the following code snippet runs, what will be printed? (3pts)

```
int i = 1;
int *p = &i;
*p = 2;
printf("%d, %d, %d\n", i, *p, (i == *p) );
```

Answer:
2, 2, 1
1pt Each.

   b. When the following code snippet runs, what will be printed? (5pts)

```
int i, j;
int *p = &j, *q = &i;
int **k = &q;
*p = 1;
*q = 2;
*k = q;
k = &q;
*k = &j;
*p = 3;
**k = 4;
printf("i: %d, j: %d\n", i, j);
```

Answer:
i: 2, j: 4
i (2pt), j (3pt)

Student ID: Name:

c. When the following code snippet runs, what will be printed? (10pts)

```
#include <stdio.h>

int main() {
      int num_arr[4][5] = {{0,1,2}, {3,4}, {5,6,7,8}, {9}};
      int *ptr3;
      printf("Hint: num_arr[1] (%p), &num_arr[1][0] (%p),
      num_arr+1 (%p) are all same!\n", num_arr[1],
      &num_arr[1][0], num_arr+1);
      ptr3 = *(num_arr + 2);
      printf("*ptr3: %d\n", *ptr3);
      printf("*(&num_arr[1][2]+1):%d\n", *(&num_arr[1][2]+1));
      printf("*(num_arr[2] + 5): %d\n", *(num_arr[2] + 5));
      printf("sizeof(num_arr) %d\n", (int)sizeof(num_arr));
      printf("sizeof(num_arr[3]) %d\n",
      (int)sizeof(num_arr[3]));
}
```

Answer:

```
Hint: num_arr[1] (0x7ffcc5fb6224), &num_arr[1][0] (0x7ffcc5fb6224),
num_arr+1 (0x7ffcc5fb6224) are all same!
*ptr3: 5
*(&num_arr[1][2]+1):0
*(num_arr[2] + 5): 9
sizeof(num_arr) 80
sizeof(num_arr[3]) 20
```

Line 1: 0pt (This line is not graded. )
Line 2: 2pt
Line 3: 2pt
Line 4: 2pt
Line 5: 2pt
Line 6: 2pt
Numbers only at each line: 0pt

Student ID:                          Name:

7. Below code is finding the substring and concatenating the two strings. Answer the questions below (10pts).

```
#include <stdio.h>

char *strcat(char *dest, const char *src)
{
        char *tmp = dest;

        while (*dest)
                dest++;
        while ((*dest++ = *src++) != '\0')
                ;
        return tmp;
}

char *strchr(const char *s, int c) {
        for (; *s != (char)c; ++s)
                if (*s == '\0')
                        return NULL;
        return (char *)s;
}

int main ()
{
        char *name1 = "Happy";
        const char *name2 = " OSLAB!";
        char *end, target = 'y';

        *end = strchr(name1, target);
        strcat(name1, name2);

        printf("Question 1 : %s %s\n", end, target);
        printf("Question 2: %s\n", name1 );
}
```

a. Fix the three bugs in the above code (5pts).
   (i) char *name1 → char name1 [100]
   (ii) *end → end
   (iii) "Question 1 : %s %s\n" → “Question 1 : %s %c\n”
   or any other possible errors
   1 answer → 1.5pts, 2 answers → 3pts, 3 answers → 5pts
b. After fixing the bug, print the output of the program (5pts).
   (iv) Question 1 : y OSLAB! y
   (v) Question 2: Happy OSLAB!

Student ID: Name:

8. Implement the linked list (10pts).

```
struct list_head {
        struct list_head *next, *prev;
};

void __list_add(struct list_head *new,
                              struct list_head *prev,
                              struct list_head *next) {
        next->prev = new;
        new->next = next;
        new->prev = prev;
        prev->next = new;
}

/**
 * list_add - add a new entry
 * @new: new entry to be added
 * @head: list head to add it after
 *
 * Insert a new entry after the specified head.
 */
void list_add(struct list_head *new, struct list_head *head) {
        /* Question (a) */
}

/**
 * list_add_tail - add a new entry
 * @new: new entry to be added
 * @head: list head to add it before
 *
 * Insert a new entry before the specified head.
 */
void list_add_tail(struct list_head *new, struct list_head *head){
       /* Question (b) */
}
```

a. Code the `list_add()` with `__list_add()` (5pts)

Answer : `__list_add(new, head, head->next);`

If the answer gives partially correct actions, give partial points (desired actions: new→next=head→next, new→prev=head, head→next=new, (head→next)→prev=new).

b. Code the `list_add_tail()` with using `__list_add()` (5pts)

Answer : `__list_add(new, head->prev, head);`

If the answer gives partially correct actions, give partial points.

Student ID: Name:

9. Dynamic Memory Allocation (9pts)
   a. Is there any potential problem in the following? If so, please identify the problem (3pts).

```
p = malloc(…);
q = malloc(…);
p = q;
free(p);
free(q)
```

Answer : the memory allocated by the first malloc() is leaked.(+3)
- Semicolon miss(+1)
- Double free(+1)
- Any potential problem worth considering(+1)

   b. Is there any potential problem in the following? If so, please identify the problem (3pts).

```
int *a; int n = 10;
a = malloc(n * sizeof(int));

for (int i = 0; i <= n; i++)
  a[i] = 0;
```

Answer : when we access a[n], unallocated memory is accessed. potential memory corruption can occur. (+3)
- Any potential problem worth considering(+1)

   c. Is there any potential problem in the following? If so, please identify the problem (3pts).

```
#include <string.h>

char *p;
int n = 10;
p = (char *)malloc(n + 1);
strcpy(p, "abc");
```

Answer : No Problem(+3)
- No initialization is required as the allocated area serves as the destination for copying
- Any potential problem worth considering(+3)