

Name:

Hemos ID:

Student ID (학번):

CSED-101 Introduction to Computing, Spring 2011

Final

	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12	Total
Your Score													
Max Score	4	11	13	6	13	6	6	9	15	5	6	6	100

- Write down your name, hemos ID, and student ID.
- There are 18 pages in this exam.
- Your answers must run correctly in C programming language without error or warning. Otherwise, your answers will be considered incorrect. For example, it is ok to put more parentheses than needed in your answer, but it will be incorrect if you put fewer parentheses than needed.
- You must write your answer on the underline => _____. Scratches outside the underline will be ignored.
- The total score is 100.
- This is a 3-hour exam.

1. (4 pts) Comparing with array, write more than 2 pros and cons of linked list each. (-1 point per incorrect answer)



2. (11 pts) Write the output of the following code segment.

```
int arr[4][3] = {{0,5,1}, {4,10,8}, {7,9,3}, {6,11,2}};
int a = 2;
int b = 4;
int * p = &a;
int * q = &b;
char *arr2[5] = {"RED", "ORANGE", "YELLOW", "GREEN", "BLUE"};








printf("Address of array: %d\n", arr);
printf("1. %d, 2. %d, 3. %d\n", arr[2][1], *(arr[3]+1), *(&arr[1][1]+1)); // (1)
printf("1. %d, 2. %d, 3. %d\n", &arr[2][1], (arr[3]+1), (&arr[1][1]+1)); // (2)

printf("%d\n", **&q); // (3)
printf("%d\n", 7 * *p / *q + 7); // (4)

printf("%s\n", *(arr2 + 1)); // (5)
printf("%c\n", *(arr2[2] + 2)); // (6)
printf("%s\n", arr2[3] + 3); // (7)
```

Output:

Address of array: 1244924

- (1) (3 pts)  _____
- (2) (3 pts)  _____
- (3) (1 pt)  _____
- (4) (1 pt)  _____
- (5) (1 pt)  _____
- (6) (1 pt)  _____
- (7) (1 pt)  _____

3. (13 pts) Answer the following questions.

3.1. (8 pts) Fill out the blanks.

```
/* (2 pts) Exchange the values by pointers */
```

```
void swap_values(int *x, int *y){
```



}

```
/* (6 pts) Sort an array of integers in an ascending order using a selection sort */
```

```
void selection_sort (int arr[], int size)
```

{

```
int smallest;
```

```
int i, j;
```

```
for(i=0; i<size-1; i++)
```

 $\{$

```
smallest = i;
```

```
for(j=i+1; j<=size-1; j++)
```

if( _____) // (2 pts)

 // (2 pts)






```
swap_values();    //(2 pts)
```

}

}

3.2. (5 pts) Write the sorting steps of this array using **insertion sort** algorithm.

23, 78, 45, 8, 32, 56

	23	78	45	8	32	56
Step 1						
Step 2						
Step 3						
Step 4						
Step 5						

4. (6 pts) Following codes are first a few lines of a program.

```
int count(double arr[], int size, double *value);  
double measure(int arr[], int *size, double *code);  
main()  
{  
    double b[10], error, volume;  
    int n[10], size, j;
```

Explain grammatical mistakes of following function calls and correct them. If there is no error to correct, just leave it blank.

- 1) (2 pts) `volume = measure(n, *size, *error);`



- 2) (2 pts) `n[7] = count(&b, n[7], &error);`



- 3) (2 pts) `error = measure(&n[0], &count(b, n[3], &volume), &volume);`



5. (13 pts) Answer the following questions.

5.1. (6 pts) We want to write a function "MyStrTok" which separates a string into two strings based on a separating character. Complete the "MyStrTok" function below. Assume that a separating character "sep" always exists in "str", thus do not consider any errors in the function. Note that the function must be written efficiently in terms of time and space. For example, it must not use unnecessary memory allocations.


```
#include <stdio.h>

char* MyStrTok(char* str, char sep);

int main()
{
    char str[10] = "153.18";
    char* str2;

    str2 = MyStrTok(str, '.'); // str gets "153" and str2 gets "18"
    printf("%s, %s\n", str, str2); // outputs "153, 18"

    return 0;
}

char* MyStrTok(char* str, char sep)
{
    
}
}
```

- 5.2. (3 pts) The following function ***MyStrCpy*** copies ***src***, including the terminating null character, to the location specified by ***dst***.

```
void MyStrCpy(char *dst, char *src)
```

```
{
```



```
}
```

- 5.3. (4 pts) The following function ***MyStrChr*** searches for the first occurrence of a character from the beginning of the string and return the pointer of it. Complete the function ***MyStrChr***(). Assume that a character "ch" always exists in "string", thus do not consider any errors in the function. (Hint, use the sequential search)

```
char* MyStrChr (const char* string, int ch)
```

```
{
```



```
}
```

6. (6 pts) What is the output of the following program?

```
#include <stdio.h>

typedef struct
{
    int i;
    int *p;
}FOO;

typedef FOO * pFOO;

int main(void)
{
    int n = 0;

    FOO foo1 = {n, &n};
    FOO foo2 = foo1;
    pFOO foo3 = &foo1;

    foo1.i = 1;
    n = 2;

    printf("%d %d\n", foo1.i, *foo1.p);
    printf("%d %d\n", foo2.i, *foo2.p);
    printf("%d %d\n", foo3->i, *foo3->p);

}
```

Output:



7. (6 pts) Write the output of the following program.

```
#include <stdio.h>

enum DAY { SAT=6, SUN=0, MON, TUE, WED, THU, FRI };
enum RAINBOW { RED, ORANGE, YELLOW=4, GREEN, BLUE, INDIGO, VIOLET };

int main() {
    enum DAY day = MON;
    enum RAINBOW color = ORANGE;

    if( day == color ) printf("MON and ORANGE are the same.\n");
    else printf("MON and ORANGE are not the same.\n");

    enum DAY i;
    for(i= SAT; i<FRI; i++)
        printf("%d", i );
    printf("\n");

    for(i= RED; i<VIOLET; i++)
        printf("%d", i );
    printf("\n");

    return 0;
}
```

Output:



8. (9 pts) Complete the code of the following program.

Command:

program.exe This is the final exam

Output:



The number of arguments: 6


Arguments: program.exe This is the final exam


Lengths of each the argument: 11 4 2 3 5 4


Arguments (reverse order): exam final the is This program.exe

```
#include <stdio.h>
#include <string.h>

int main(){ // (1) get arguments
    int i;
    // (2) print the number of arguments
    printf("The number of arguments: %d\n", );

    // (3) print the arguments
    printf("Arguments: ");
    
    printf("\n");

    // (4) print the lengths of each the argument
    printf("Lengths of each the argument: ");
    
    printf("\n");

    // (5) print the arguments (reverse order)
    printf("Arguments (reverse order): ");
    
    printf("\n");

    return 0;
}
```

9. (15 pts) There is a file which contains information about 100 students. The name of the file is "student_info.dat" and the file was created by following program.

```
#include <stdio.h>

typedef struct
{
    char name[20];
    char gender[7];
    unsigned int studentID;
    unsigned int score;
}STUDENT;

void main()
{
    STUDENT student[100];
    FILE *fp;
    ...

    fp = fopen("student_info.dat", "wb");
    fwrite(student, sizeof(STUDENT), 100, fp);
    fclose(fp);
}
```



Refer to above codes to complete following program. This program reads student information from "student_info.dat" file and modifies every female student's score as 100. This modification should be applied to "student_info.dat" file. The *gender* is either "Male" or "Female".

<Caution!!! Don't use additional variables and don't use *fopen()* function more than once!!!>

```
#include <stdio.h>
#include <string.h>

typedef struct
{
    char name[20];
    char gender[7];
    unsigned int studentID;
    unsigned int score;
}STUDENT;
```

```

void main()
{

    STUDENT student;
    FILE *fp;
    int i;

    _____

    _____

    for(i=0;i<100;i++)
    {
        fread(_____);

        if(_____)
        {
            student.score=100;

            _____

            _____

        }
    }

    _____

}

```

10. (5 pts) Complete the following "BitwiseEqual" function using only bitwise operators. The function returns a 16 bits such that each bit is one if the corresponding two bits of the input bits are equal, or zero if they are different. For example, if the function has arguments of two 16bits "0000 0000 0000 0000" and "0000 0000 1111 1111", then the returning bits is "1111 1111 0000 0000". Accordingly, the main function below prints "OxFF00" as result.

```
#include <stdio.h>
#include <stdint.h>

uint16_t BitwiseEqual(uint16_t a, uint16_t b);

int main()
{
    uint16_t a = 0x0000;
    uint16_t b = 0x00FF;
    uint16_t c;

    c = BitwiseEqual(a, b);
    printf("%#16X\n", c);

    return 0;
}

uint16_t BitwiseEqual(uint16_t a, uint16_t b)
{



}
```

11. (6 pts) The following "deleteNode" function deletes the node pointed by "pCur" in the list pointed by "pList". "pPre" points the previous node of the node pointed by "pCur".

```
NODE* deleteNode(NODE* pList, NODE* pPre, NODE* pCur)
{
    if (pPre == NULL)
        pList = pCur->link;
    else
        pPre->link = pCur->link;
    free(pCur);
    return pList;
}
```

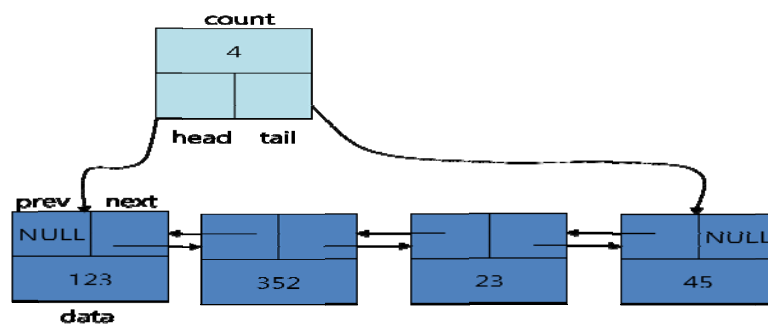
Modify the function without "pCur", that is, write the following "deleteNodeWithPre" function with only "pPre" as argument which points the previous node of the node that we want to delete. Assume that "pPre" is assigned with NULL if it is the first node that we want to delete.

```
NODE* deleteNodeWithPre(NODE* pList, NODE* pPre)
{
    NODE* pCur;

    if (pPre == NULL)
    {
        
    }
    else
    {
        
    }

    free(pCur);
    return pList;
}
```

12. (6 pts) The following is the structure with doubly linked list containing integer data. Fill out blanks.



```
typedef struct element {
    int data;
    struct element *prev, *next;    // prev and next are initialized as NULL when they are created.
} Element;
```

```
typedef struct list {
    int count;
    Element *head, *tail;          // head and tail are initialized as NULL when they are created.
} List;
```

// delete index-th element from list li. the index range is from 0 to count-1.

```
void remove_data(List *li, int index) {
```

```
    Element *current;
```

```
    int i;
```

```
    current = li->head;
```

```
    for(i=0; i<index; i++) {
```





```
        current = current->next;
```

```
    }
```

```
    if(current==li->head) {
```



```
    }
```

```
else if(current == li->tail) {  
    li->tail = current->prev;  
    li->tail->next = NULL;  
}  
else {  
    current->  ->next =  _____;  
    current->  ->prev =  _____;  
}  
free(current);  
li->count--;  
}
```

Appendix

A. FILE 입출력 관련 함수

- FILE* fopen(const char* filename, const char* mode);
 - ⊙ Same as text I/O, but *mode* is different
 - ⊙ Mode
 - ⊙ b: binary indicator
 - ⊙ Six binary modes: read binary(rb), write binary(wb), append binary(ab), read and update binary(r+b), write and update binary(w+b), and append and update binary (a+b)
- int fclose(FILE *fp);
 - ⊙ Same as text I/O
- int fread(void *pInArea, int elementSize, int count, FILE *sp);
 - ⊙ Reads up to *count* objects, each *elementSize* bytes long, from input file *sp*, storing them in the memory pointed to by *pInArea*
- int fwrite(void *pOutArea, int elementSize, int count, FILE* sp);
 - ⊙ Writes up to *count* objects, each *elementSize* bytes long, from the memory pointed to by *pOutArea* to the output file *sp*
- void rewind(FILE* stream);
 - ⊙ Sets the file position indicator for *stream* to the beginning of the file
- long int ftell(FILE* stream);
 - ⊙ Returns the current file position for *stream*
 - ⊙ Returned value is the number of bytes from the beginning of the file to the current file position
- int fseek(FILE* stream, long offset, int wherefrom);
 - ⊙ Sets the file position indicator for *stream*
 - ⊙ New byte position is obtained by adding *offset* to the position specified by *wherefrom*
 - ⊙ *wherefrom*
 - ⊙ SEEK_CUR: The offset is computed from the current position in the file
 - ⊙ SEEK_SET: The offset is computed from the beginning of the file
 - ⊙ SEEK_END: The offset is computed from the end of the file
- int feof(FILE* stream);
 - ⊙ Checks the end-of-file indicator for *stream* and returns non-zero if it is at the end
- int ferror(FILE* stream);
 - ⊙ Checks the error indicator for *stream* and returns non-zero if an error has occurred
- void clearerr(FILE* stream);
 - ⊙ Clears the end-of-file and error indicator for *stream*

연습지

연습지