

Course Code	CSC 215			
Course Title	Procedural Programming			
Section No.				
Semester	Semester 2: 2022-2023 (442)			
Exam	Final Exam			
Date	20/02/2023	Duration	120 minutes	
Student Name				
Student ID				

Questions	1	2	3	4	5	Total Grade
Full mark	10	10	5	5	10	40
Student's mark						

Instructions:

- This exam has a total of 40 marks.
- Write clearly and neatly.
- Copy your answers to questions 11 to 20 in the table.
- For all questions, assume the size of the integer type and the address is 32-bits.
- Assume standard library header files are included where needed.

Feedback/Comments:

10 marks

Question1: Write **T** next to the true statement, and **F** next to the False one:

Statement		True/False
1	The ++ operator has a higher precedence than the * operator in C, so ++x * 3 is equivalent to (x++) * 3.	
2	double has limited precision and can only represent a finite set of decimal numbers with some rounding error.	
3	In C, the switch statement can only be used with integer and character types.	
4	In C, the static keyword can be used to declare a variable that retains its value between function calls.	
5	It is possible to use pointer arithmetic to access memory locations that are outside the allocated memory block for a pointer variable.	
6	Given: struct V{char s[3];union{float f;int i[3];} d;}; the result of sizeof(struct V) is 16 bytes.	
7	A struct in C programming is a user-defined data type that allows you to store different data types in the same memory location.	
8	fgets function in C always reads in exactly the number of characters specified by the second argument.	
9	An example of the declaration of function pointer: int(*fp)(int,char*) =0; is valid.	
10	The time complexity of random access to an element in a linked list of size n, is O(n).	

10 marks

Question 2: Copy your answer for each of the following questions to the table:

1	2	3	4	5	6	7	8	9	10

1. What is the output of the following code segment (if any)?

```
int main() {
    typedef int a;
    a b=2, c=8, d;
    d =(b*2)/2+8;
    printf("%d",d);
    return 0;}
```

A. 8 B. 10 C. 16 D. Compilation error

2. What is the output of the following code segment (if any)?

```
const int MAX = 3;
int main (){
    int var[] = {10, 100, 200}, i, *ptr;
    ptr = var;
    for ( i = 0; i < MAX; i++) printf("%d ", (*ptr)++);
    return 0;}
```

B. 10 100 200 B. 10 11 12 C. Address of array var D. Compilation error

3. Which data structure is used to handle recursion in C?

- A. Stack B. Queue C. Tree D. Linked List
-

4. What is the output of the following code segment (if any)?

```
int main(){
    int arr[] = {1, 2, 3, 4, 5}, *ptr1 = arr, *ptr2 = &arr[4];
    while (ptr1 < ptr2){
        int temp = *ptr1;
        *ptr1 = *ptr2;
        *ptr2 = temp;
        ptr1++; ptr2--;
    }
    for (int i = 0; i < 5; i++) printf("%d ", arr[i]);
    return 0; }
```

- A. 1 2 3 4 5 B. 5 4 3 2 1 C. 4 2 1 3 5 D. Compilation error
-

5. Which is the equivalent pointer expression for a[i][j] ?

```
struct School {
    int age; short rollNo;
};
void solve(){
    struct School sc;
    sc.age = 19;
    sc.rollNo = 82;
    printf("%d", (int)sizeof(sc));
}
int main() {
    solve();
    return 0;
}
```

- A. 4 B. 6 C. 8 D. 16
-

6. What is the output of the following code segment (if any)?

```
int n=4, m=3, **A, i;
if ((A = (int**)malloc(n*sizeof(int*))))
    if ((A[0] = (int*)calloc(n*m, sizeof(int))))
        for (i = 1; i<n; i++) A[i] = A[0] + m*i;
printf("%d\n", A[3]-A[2]);
```

- A. 3 B. 12 C. 10 D. 5
-

7. Which of the following declarations is valid?

```
void foo(int* p){printf("%d\n", *p); }
int main(){
    int i = 10, *p = &i;
    foo(p++);
    return 0; }
```

- A. 10 B. 11 C. Undefined value D. Runtime error
-

8. What is the content of `file.c` after executing the following segment of code?

```
FILE* fp=fopen("file.c", "w");
fputs("A B C D E", fp);
fseek(fp,3,SEEK_SET);
fputs("X Y Z", fp);
fclose(fp);
```

- A. A B X Y Z E B. A X Y Z E C. X Y Z D. A B C X Y Z

9. What is true about `void*` ?

- A. Does not indicate a specific pointer type
 B. Very useful when you want a pointer to point to data of different types at different times.
 C. It can be casted to any pointer type.
 D. All of the given

10. If `start` points to the first node in the following linked list:



What will be the contents of the list after `fun(start, 0);` is called?

```
void fun(Node *head, Node *end){
    Node *p = head, *q = head;
    if (!head || head==end || (end && end->next == head)) return;
    while((q->next != end)) q = q->next;
    int temp = p->data;
    p->data = q->data;
    q->data = temp;
    fun(head->next, q);
}
```

- A. 1 2 3 4 5 6 B. 6 5 4 3 2 1 C. 2 1 4 3 6 5 D. 1 3 2 5 4 6

Question 3: Consider the following declaration and its memory representation.

```
int a[5] ;
int* p = malloc(5*sizeof(int));
```

Determine the value of each statement in the table below.

Address	a:0x20000	0x20004	0x20008	0x2000C	0x20010	p:0x20014	0x30010	0x30014	0x30018	0x3001C	0x30020
Value						0x30010					

Evaluate each of the following expressions:

Expression	a	&a	&a+1	p	&p	&p+1
Value						

Expression	(char*) &a[3] - (char*) &a[1]	(short*) (p+3) - (short*) (p+1)
Value		

Question 4: Consider the following segments of code. Determine the valid and invalid statements. Suppose each statement is independent.

- A.** `const char c = 'a'; const char *ptr; ptr = &c;`
`c = 'b';` ☐ valid ☐ invalid
`*ptr = 'b';` ☐ valid ☐ invalid
- B.** `void first(){printf("Final");}`
`int main(){`
`void (*ptr)() = first;`
`ptr();` ☐ valid ☐ invalid
`(*ptr)();` ☐ valid ☐ invalid
`return 0; }`
- C.** `struct person {`
`int age;`
`int kidsAge[3];`
`char fname[10] = "Fahad";` ☐ valid ☐ invalid
`} p;`
`int main(){`
`struct person kidsAge[] = {10,15};` ☐ valid ☐ invalid
`printf("%d",p.kidsAge[2]);`
`return 0; }`
- D.** `struct employee{`
`char* name;`
`int id;`
`float salary;`
`struct employee achievements; };` ☐ valid ☐ invalid
`struct employee *e;` ☐ valid ☐ invalid
- E.** `int main(){`
`int x=5;`
`void* px=&x;`
`printf("%d",*px); return 0;}` ☐ valid ☐ invalid
- F.** `enum State {small=1, medium=0, large };` ☐ valid ☐ invalid
`int main() {`
`printf("%d, %d, %d", small, medium, large); return 0;}`
- G.** `int *p[2];`
`int** ptr=(int*)malloc(10*sizeof(int));` ☐ valid ☐ invalid
`p[0] = (int*)malloc(5*sizeof(int));`
`p[1] = (int*)malloc(5*sizeof(int));`

Question 5: Complete the program that reads data from a file and stores it in linked lists in ascending order. Each file line contains a list of integers separated by commas, and represents a new linked list. For example, the file might look like this:

data.txt
1,3,2,4,5
9,7,8,6
10,11,12,13,14

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
struct node {int data; struct node* next; };
```

A. This function prints out the data in each node of the list that starts at head.

```
void print_list(struct node* head) {
```

```
.....
.....
.....
printf("\n");
```

```
}
```

B. This function inserts a new node with the value data in ascending order into the linked list.

```
void insert(struct node** head, int data) {
```

```
// allocate memory for new node
```

```
struct node* new_node = .....;
```

```
new_node->data = data;
```

```
// Case 1: Empty list or new node goes at the beginning
```

```
if (.....) {
```

```
new_node->next = *head;
```

```
*head = new_node;
```

```
return;
```

```
}
```

```
// Case 2: New node goes in the middle or at the end
```

```
struct node* current = *head;
```

```
while (.....)
```

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

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

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

```
}
```

C. This function reads each line from the file, converts the comma-separated integers in each line into a linkedlist, inserts the integers in ascending order, and prints the resulting linkedlist.

```
int main() {
```

```
char line[1024], *token;
```

```
struct node* head = NULL;
```

```
// open the file in read mode
```

```
.....;
```

```

    if (.....) { // check if the file was
opened
    // read each line from the file

    while (.....) {

        token = strtok(.....); // get the first token

        while (.....) { // loop over all tokens

            int data = .....; // convert token to
integer

            .....; // insert integer into list

            token = .....; // get next token in line
        }

        .....; // print contents of linked list

        .....; // reset the linked list for the next line
    }

    .....; // close the file
}
return 0;
}

```