National University of Computer & Emerging Sciences, Karachi
Spring 2023 CS-Department
Final Examination
22nd May 2023, 08:30 AM – 11:30 AM

Course Code: CS-1002	**Course Name: Programming Fundamentals**
Instructor Name: Mr. Basit Ali, Mr. Shoaib Rauf, Mr. Ali Fatmi	
Student Roll No:	**Section No:**

Instructions:

- Please return the question paper.
- Please read each question completely before answering it. There are **5 questions and 5 pages**
- In case of any ambiguity, you may make assumption. However, your assumption should not contradict any statement in the question paper.
- The exam is self-explanatory. Show all steps clearly.

Time: 180 minutes. **Max Marks**: 100 points

Question 01: Multiple Choice Questions (MCQs) (CLO1) **20 points**

1. What is the size of a C structure?
   a) C structure is always 128 bytes.
   b) Size of C structure is the total bytes of all elements of structure.
   c) Size of C structure is the size of largest element.
   d) None of the above

2. Choose a correct statement about C structures.
   a) Structure elements can be initialized at the time of declaration.
   b) Structure members cannot be initialized at the time of declaration
   c) Only integer members of structure can be initialized at the time of declaration
   d) None of the above

3. Choose a correct statement about C structure.

```
int main()
{
    struct ship
    {

    };
    return 0;
}
```

   a) It is wrong to define an empty structure
   b) Member variables can be added to a structure even after its first definition.
   c) There is no use of defining an empty structure
   d) None of the above

4. Choose a correct statement about C structure elements.
   a) Structure elements are stored on random free memory locations
   b) structure elements are stored in register memory locations
   c) structure elements are stored in contiguous memory locations
   d) None of the above.

5. Array is an example of _______ type memory allocation.
   a) Compile time
   b) Run time
   c) Both A and B
   d) None of the above

6. The parameter passing mechanism for an array is
   a) call by value
   b) call by reference
   c) call by value-result
   d) None of the above

7. Which of the following function sets first n characters of a string to a given character?
   a) strset()
   b) strnset()
   c) strcset()
   d) strinit()

8. Which of the following gives the memory address of the first element in array foo, an array with 10 elements?
   a) foo
   b) &foo
   c) foo[0]
   d) &foo[0]

9. What is the output of this program?

```
#include <stdio.h>
void main(){
char c[] = "GATE2011";
char *p =c;
printf("%s", p + p[3] - p[1]);
}
```

   a) GATE2011
   b) E2011
   c) 2011
   d) 11

10. How do you dynamically allocate memory for an array of structures in C?
    a) structArray = (struct StructureType*)malloc(sizeof(struct StructureType) * numElements);
    b) structArray = (struct StructureType*)malloc(sizeof(struct StructureType));
    c) structArray = malloc(struct StructureType * numElements);
    d) structArray = malloc(sizeof(struct StructureType) * numElements);

11. In C, what is the purpose of the "fgets()" function when dealing with file input?
    a) It reads a single character from a file
    b) It reads a line of text from a file
    c) It writes formatted data to a file
    d) It appends data to an existing file

12. Which of the following is true about the "sizeof" operator in C?
    a) It returns the size of a structure excluding padding bytes
    b) It can be used to find the size of dynamically allocated memory

c) It can be used to find the number of elements in an array
d) It returns the size of a structure including padding bytes

13. Which of the following statements is true about nested structures in C?
a) A structure can only be nested within another structure if they have the same member names.
b) Nesting structures increases memory consumption and should be avoided.
c) Nested structures can have their own functions defined within the outer structure.
d) Accessing members of a nested structure requires using multiple levels of indirection.

14. What is the purpose of the "->" operator in C when working with structures?
a) It is used to access members of a structure when using a pointer to the structure.
b) It is used to access members of a structure when using the structure directly.
c) It is used to compare two structures for equality.
d) It is used to assign a value to a structure member.

15. What is the correct way to declare a pointer to a constant integer in C?
a) int *const ptr;
b) const int *ptr;
c) const *int ptr;
d) const int *const ptr;

16. What is the output of the following code snippet?

```
int arr[5] = {1, 2, 3, 4, 5};
int *ptr = &arr[2];
printf("%d", *ptr++);
```

a) 2
b) 3
c) 4
d) 5

17. What does the following code snippet do?

```
char *str = "Hello";
*str = 'h';
printf("%s", str);
```

a) Prints "Hello"
b) Prints "hello"
c) Causes a runtime error
d) None of the above

18. What is the output of the following code snippet?

```
int arr[2][2] = {{1, 2}, {3, 4}};
int *ptr = (int *)arr;
printf("%d", *(ptr + 3));
```

a) 1
b) 2
c) 3
d) 4

19. What is the correct way to declare a pointer to a 2D array in C?
a) int **ptr;
b) int *ptr[2][2];
c) int (*ptr)[2];
d) int **ptr[2][2];

20. What is the output of the following code snippet?

```
int arr[3][3] = {
    {1, 2, 3},
    {4, 5, 6},
    {7, 8, 9}
};
int (*ptr)[3] = arr;
printf("%d", *(*ptr + 2));
```

a) 1
b) 2
[✓ c) 3]
d) 4

Question 02: Short Programs (CLO2) **30 points**

a) Write a C program to print sum of the non-repeated elements of an array.

Sample Input: 1, 2, 3, 2, 2, 5, 6, 1
Sample Output: 3 + 5 + 6 = 14

```
int main() {
   int arr[] = {1, 2, 3, 2, 2, 5, 6, 1};
   int size = sizeof(arr) / sizeof(arr[0]);
   int sum = 0;
          int i,j;
   for ( i = 0; i < size; i++) {
      int count = 0;
      for (j = 0; j < size; j++) {
         if (arr[i] == arr[j]) {
            count++;
         }
      }
      if (count == 1) {
         sum += arr[i];
      }
   }

   printf("Sum of non-repeated elements: %d\n", sum);

   return 0;
}
```

b) Write a recursive function ***"int Recur(int)"*** to find the sum of digits of a positive integer.

Sample Input: 1234
Sample Output: 10

```
#include <stdio.h>

int Recur(int num) {
   if (num == 0) {
      return 0;  // Base case: If the number is 0, return 0.
   } else {
      return (num % 10) + Recur(num / 10);  // Recursive case: Add the last digit to the sum and
call Recur on the remaining digits.
   }
}
```

```
int main() {
   int number = 1234;
   int sum = Recur(number);

   printf("Sum of digits: %d\n", sum);

   return 0;
}
```

c) Write a C program to find the frequency of every word in the given string

Sample Input:
Enter the string: this is pak it is pak
Sample Output:
```
Frequency of 'this' is: 1
Frequency of 'is' is: 2
Frequency of 'pak' is: 2
Frequency of 'it' is: 1
```

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

void wordFrequency(char *str) {
   const char *delimiters = " \t\n\r\f\v";  // Delimiters for strtok function
   char *token;
   int count = 0;
   char words[100][100];
   int frequencies[100] = {0};
   int totalWords = 0;

   // Tokenize the string into words
   token = strtok(str, delimiters);
   while (token != NULL) {
      strcpy(words[count], token);
      count++;
      token = strtok(NULL, delimiters);
   }
   totalWords = count;

   // Count the frequency of each word
   int i,j;
   for (i = 0; i < totalWords; i++) {
      if (frequencies[i] == -1) {
         continue;
      }
      for (j = i + 1; j < totalWords; j++) {
         if (strcmp(words[i], words[j]) == 0) {
            frequencies[i]++;
            frequencies[j] = -1;  // Mark the repeated word as -1 to skip it in further iterations
         }
      }
   }

   // Print the frequency of each word
   for (i = 0; i < totalWords; i++) {
      if (frequencies[i] != -1) {
         printf("Frequency of '%s' is: %d\n", words[i], frequencies[i] + 1);
      }
   }
}

int main() {
   char str[1000];

   printf("Enter the string: ");
   fgets(str, sizeof(str), stdin);

   // Remove the newline character from the input
   if (str[strlen(str) - 1] == '\n') {
      str[strlen(str) - 1] = '\0';
   }

   wordFrequency(str);

   return 0;
}
```

Question 03:	(CLO3) **20 points**

Write a C program that reads data from a text file called **'books.txt'** and stores it into a structure called 'Book'. The structure should contain the following members: 'title' (string), 'author' (string), and 'price' (float). Implement a function ***"void sortBooksByPrice(struct Book books[], int numBooks)"*** that takes an array of Book structures and the number of books as parameters. The function should sort the books in ascending order based on their prices using any sorting algorithm of your choice. In the main function, read data from the file into an array of Book structures, call the 'sortBooksByPrice' function to sort the books, and then display the sorted list of books.

Here is an example of how the 'books.txt' file could look like:

```
Title: Book1
Author: Author1
Price: 10.5

Title: Book2
Author: Author2
Price: 8.99

Title: Book3
Author: Author3
Price: 12.75
```

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

#define MAX_TITLE_LENGTH 100
#define MAX_AUTHOR_LENGTH 100
#define MAX_BOOKS 100

struct Book {
    char title[MAX_TITLE_LENGTH];
    char author[MAX_AUTHOR_LENGTH];
    float price;
};
```

```
void sortBooksByPrice(struct Book books[], int numBooks) {
    // Bubble Sort algorithm used for simplicity
    for (int i = 0; i < numBooks - 1; i++) {
        for (int j = 0; j < numBooks - i - 1; j++) {
            if (books[j].price > books[j + 1].price) {
                // Swap books
                struct Book temp = books[j];
                books[j] = books[j + 1];
                books[j + 1] = temp;
            }
        }
    }
}

int main() {
    FILE *file = fopen("books.txt", "r");
    if (file == NULL) {
        printf("Failed to open the file.\n");
        return 1;
    }

    struct Book books[MAX_BOOKS];
    int numBooks = 0;

    char line[100];
    while (fgets(line, sizeof(line), file)) {
        if (strncmp(line, "Title: ", 7) == 0) {
            strcpy(books[numBooks].title, line + 7);
            books[numBooks].title[strcspn(books[numBooks].title, "\n")] = '\0'; // Remove the newline
character
        } else if (strncmp(line, "Author: ", 8) == 0) {
            strcpy(books[numBooks].author, line + 8);
            books[numBooks].author[strcspn(books[numBooks].author, "\n")] = '\0'; // Remove the newline
character
        } else if (strncmp(line, "Price: ", 7) == 0) {
            sscanf(line + 7, "%f", &books[numBooks].price);
            numBooks++;
        }
    }

    fclose(file);

    sortBooksByPrice(books, numBooks);

    printf("Sorted Books:\n");
    for (int i = 0; i < numBooks; i++) {
        printf("Title: %s\n", books[i].title);
        printf("Author: %s\n", books[i].author);
        printf("Price: %.2f\n\n", books[i].price);
    }

    return 0;
}
```

Question 04:	(CLO3) **20 points**

Suppose you have a passage of 400 to 500 characters based on user input. You need to write a program that eliminates the stop words from that passage. You need to take the input of stop words from the user also. After eliminating these stop words, the program should dynamically resize the passage using DMA and display the resized passage.

Ensure that the program handles memory allocation and deallocation appropriately to avoid memory leaks.

Sample Input:

Passage = “Hi, My name is Riaz Channa. I work in FAST-NUCES”

Stop words = {“is“, “in”}

Sample Output:

Passage = “Hi, My name Riaz Channa, I work FAST-NUCES”

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

#define INITIAL_CAPACITY 500

char* removeStopWords(const char* passage, const char** stopWords, int numStopWords) {
   int passageLength = strlen(passage);
   char* resizedPassage = malloc(sizeof(char) * (passageLength + 1)); // Allocate memory for resized
passage
   int resizedIndex = 0;

   int start = 0;
   int end = 0;

   while (end <= passageLength) {
      if (passage[end] == ' ' || passage[end] == '\0') {
         int wordLength = end - start;
         int isStopWord = 0;
          int i;
         for (i = 0; i < numStopWords; i++) {
            if  (wordLength  ==  strlen(stopWords[i])  &&  strncmp(&passage[start],  stopWords[i],
wordLength) == 0) {
               isStopWord = 1;
               break;
            }
         }

         if (!isStopWord) {
            strncpy(&resizedPassage[resizedIndex], &passage[start], wordLength);
            resizedIndex += wordLength;
            resizedPassage[resizedIndex] = ' ';
            resizedIndex++;
         }

         start = end + 1;
      }

      end++;
   }

   resizedPassage[resizedIndex - 1] = '\0'; // Remove the trailing space

   return resizedPassage;
}

int main() {
   char passage[500];
   printf("Enter the passage (400-500 characters):\n");
   fgets(passage, sizeof(passage), stdin);
```

```
    passage[strcspn(passage, "\n")] = '\0'; // Remove the newline character

    int numStopWords;
    printf("Enter the number of stop words: ");
    scanf("%d", &numStopWords);
    getchar(); // Clear the newline character from the input buffer

    const char** stopWords = malloc(sizeof(char*) * numStopWords); // Allocate memory for stop words

    printf("Enter the stop words:\n");
    int i;
    for (i = 0; i < numStopWords; i++) {
        stopWords[i] = malloc(sizeof(char) * 20); // Assuming each stop word has a maximum length of 19
characters
        fgets(stopWords[i], 20, stdin);
        stopWords[i][strcspn(stopWords[i], "\n")] = '\0'; // Remove the newline character
    }

    char* resizedPassage = removeStopWords(passage, stopWords, numStopWords);

    printf("\nResized Passage: %s\n", resizedPassage);

    // Free memory
    for (int i = 0; i < numStopWords; i++) {
        free((void*)stopWords[i]);
    }
    free(stopWords);
    free(resizedPassage);

    return 0;
}
```

Question 05:	(CLO2) **10 points**

Consider two Structures "Student" and "Marks" with the below prototype.

Student:
{Student_ID, Student_Name, Student_Email, Exam_ID}

Exam:
{Course1, Course2, Course3, Course4, Course5, Course6, Exam_ID}

You are required to perform the following operations:

a. Use loops to initialize both arrays of 100 students and their marks in six courses respectively.
b. Write a function ***"bool report_card(struct student s[100], struct Exam e[100])"*** that determine whether the student passed or failed in exam. Please note that the total marks of each course is 100 and passing criterion is achieving at least 50% marks in all courses.

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

#define NUM_STUDENTS 100
#define NUM_COURSES 6

struct Student {
    int Student_ID;
    char Student_Name[50];
    char Student_Email[50];
    int Exam_ID;
};
```

```
struct Exam {
    int Course1;
    int Course2;
    int Course3;
    int Course4;
    int Course5;
    int Course6;
    int Exam_ID;
};

void initializeStudents(struct Student students[]) {
    for (int i = 0; i < NUM_STUDENTS; i++) {
        students[i].Student_ID = i + 1;
        sprintf(students[i].Student_Name, "Student %d", i + 1);
        sprintf(students[i].Student_Email, "student%d@example.com", i + 1);
        students[i].Exam_ID = i + 1;
    }
}

void initializeMarks(struct Exam exams[]) {
    for (int i = 0; i < NUM_STUDENTS; i++) {
        exams[i].Course1 = rand() % 101; // Generate a random mark between 0 and 100
        exams[i].Course2 = rand() % 101;
        exams[i].Course3 = rand() % 101;
        exams[i].Course4 = rand() % 101;
        exams[i].Course5 = rand() % 101;
        exams[i].Course6 = rand() % 101;
        exams[i].Exam_ID = i + 1;
    }
}

bool report_card(struct Student students[], struct Exam exams[]) {
    bool passed = true;

    for (int i = 0; i < NUM_STUDENTS; i++) {
        if ((exams[i].Course1 < 50) || (exams[i].Course2 < 50) || (exams[i].Course3 < 50) ||
            (exams[i].Course4 < 50) || (exams[i].Course5 < 50) || (exams[i].Course6 < 50)) {
            passed = false;
            printf("Student ID: %d - Failed\n", students[i].Student_ID);
        } else {
            printf("Student ID: %d - Passed\n", students[i].Student_ID);
        }
    }

    return passed;
}

int main() {
    struct Student students[NUM_STUDENTS];
    struct Exam exams[NUM_STUDENTS];

    initializeStudents(students);
    initializeMarks(exams);

    bool allPassed = report_card(students, exams);

    if (allPassed) {
        printf("All students passed the exam.\n");
    } else {
        printf("Some students failed the exam.\n");
    }

    return 0;
}
```

******** Good Luck ********