**Internal Assessment (Assignment)**

**Name:** Deepankar Sharma

**Student Id:** 233512013

**Course Code:** OMC103

**Course Title:** Programming and Problem-Solving

**Assignment No.: 2**

**Part A** **(10 1 = 10 Marks)**

|  |  |  |
| --- | --- | --- |
| **MCQ No.** | **Question** | |
| **1** | The size of a structure is--------- | |
| Answer Choices: | 1. 1 byte 2. Total bytes of all structure members 3. 4 bytes 4. 2 bytes | |
| **2** | Structure in C Programming is----- | |
| Answer Choices: | 1. Collection of elements of the same data type 2. Collection of elements of the different data types 3. Set of values 4. Built-in data type | |
| **3** | The maximum number of dimensions in an array is---- | |
| Answer Choices: | 1. 1 2. 2 3. 3 4. No limit | |
| **4** | When an array is passed to a function, it is interpreted as ---- | |
| Answer Choices: | 1. Address of an array 2. Address of First Element 3. Values of First Element 4. Number of elements in an array | |
| **5** | What is the value of the x in the following statement?  X = strcmp(string1, string2); | |
| Answer Choices: | 1. 1 2. False 3. 0 4. String1 | |
| **6** | | int a[5] = {1,2,3,4,5}, what is the value of a[7]? |
| Answer Keys: | | 1. 0 2. Garbage Value 3. 5 4. 1 |
| **7** | | The format specifier used for printing a string is ----- |
| Answer Keys: | | 1. %c 2. %d 3. %s 4. %f |
| **8** | | If a file opening is failed, then fopen will return---- |
| Answer Keys: | | 1. null 2. eof() 3. Depends on Compiler 4. zero |
| **9** | | The fastest loop in C programming is ------------------. |
| Answer Keys: | | 1. while 2. do-while 3. for 4. All the options |
| **10** | | What will be the output of the following program?  #include <stdio.h>  int main(){  FILE \*fp;  char \*str;  fp=fopen("demo.txt","r");// demo.txt //:First Semester MCA  while(fgets(str,6,fp)!=NULL)  puts(str);  fclose(fp);  return 0;  } |
| Answer Key | | 1. First Semester MCA 2. First S 3. First Semester 4. First Se |

**Part B (5 4 = 20 Marks)**

|  |  |
| --- | --- |
| **Q No.** | **Question** |
| **2** | Illustrate different file-handling modes using a suitable C program.  #include <stdio.h>  *int* main() {      FILE \*file;      // Open file in write mode (w)      file = fopen("example.txt", "w");      if (file == NULL) {          printf("Error opening file.\n");          return 1;      }      fprintf(file, "Hello, World!\n");      fclose(file);      // Open file in read mode (r)      file = fopen("example.txt", "r");      if (file == NULL) {          printf("Error opening file.\n");          return 1;      }  *char* line[100];      while (fgets(line, sizeof(line), file)) {          printf("%s", line);      }  fclose(file);      // Open file in append mode (a)      file = fopen("example.txt", "a");      if (file == NULL) {          printf("Error opening file.\n");          return 1;      }      fprintf(file, "Appending this line.\n");      fclose(file);      return 0;  } |
| **3** | Design and develop a C program to read a text and count the number of alphabets, spaces, and digits.  #include <stdio.h>  *int* main() {      FILE \*file;  *char* ch;  *int* digits = 0, spaces = 0, alphabets = 0;      file = fopen("b3\_file.txt", "r");      if (file == NULL) {          printf("Error opening file.\n");          return 1;  }      while ((ch = fgetc(file)) != EOF) {          if(ch >= '0' && ch <= '9')              digits++;          else if(ch == ' ')              spaces++;          else if((ch >= 'a' && ch <= 'z') || (ch >= 'A' && ch <= 'Z'))              alphabets++;  }      printf("Number of Alphabets: %d\n", alphabets);      printf("Number of Digits: %d\n", digits);  printf("Number of Spaces: %d\n", spaces);      fclose(file);      return 0;  } |
| **4** | Write a C program to add two complex numbers using structures. Use a concept of structures for multiple records of complex numbers.  #include <stdio.h>  typedef *struct* Complex {  *float* real;  *float* imag;  } Complex;  Complex add(Complex *n1*, Complex *n2*) {      Complex temp;      temp.real = *n1*.real + *n2*.real;      temp.imag = *n1*.imag + *n2*.imag;      return temp;  }  *int* main() {      Complex n1, n2, result;      printf("For 1st complex number \n");      printf("Enter real and imaginary part respectively:\n");      scanf("%f %f", &n1.real, &n1.imag);      printf("\nFor 2nd complex number \n");      printf("Enter real and imaginary part respectively:\n");      scanf("%f %f", &n2.real, &n2.imag);      result = add(n1, n2);      printf("Sum = %.1f + %.1fi", result.real, result.imag);      return 0;  } |
| **6** | Write the advantages and drawbacks of recursion. Write a c-program using the recursive function for Binary to Decimal Conversion.  #include <stdio.h>  #include <math.h>  *int* binaryToDecimal(*int* *binary*, *int* *n*) {      if(*binary* == 0) {          return 0;      } else {          return ((*binary*%10) \* pow(2,*n*)) + binaryToDecimal(*binary*/10, *n*+1);      }  }  *int* main() {  *int* binary;      printf("Enter a binary number: ");      scanf("%d", &binary);      printf("Decimal: %d", binaryToDecimal(binary, 0));      return 0;  } |
| **7** | Write a c-program using structures to read, write, compute average -  marks and display the students scoring above and below the average  marks for a class of N students  #include <stdio.h>  typedef *struct* {  *char* name[50];  *int* marks;  } Student;  *int* main() {  *int* i, n, sum = 0;  *float* average;      printf("Enter the number of students: ");      scanf("%d", &n);      Student students[n];      for(i = 0; i < n; i++) {          printf("Enter name and marks for student %d: ", i+1);          scanf("%s %d", students[i].name, &students[i].marks);          sum += students[i].marks;      }      average = (*float*)sum / n;      printf("\nAverage marks = %.2f\n", average);      printf("\nStudents scoring above the average:\n");      for(i = 0; i < n; i++) {          if(students[i].marks > average) {              printf("%s\n", students[i].name);          }      }      printf("\nStudents scoring below the average:\n");      for(i = 0; i < n; i++) {          if(students[i].marks < average) {              printf("%s\n", students[i].name);          }      }      return 0;  } |