National University of Computer and Emerging Sciences

FAST School of Computing **Spring-2024** **Islamabad Campus**

CS-1002: Programming Fundamentals

Final Exam

Total Time: 3 Hours
Total Marks: 100

Date: 20th May, 2024

Course Instructor(s)

Asif Malik

Student Name	Roll No.	Course Section	Student Signature

Instructions:

1. **Attempt first two questions on the question paper and return the question paper along with the answer sheet.**
2. **All solution codes should be indented as well as appropriately commented.**
3. Use permanent ink pens only. Any part done using a soft pencil will not be marked and cannot be claimed for rechecking.
4. Ensure that you do not have any electronic gadgets (like mobile phones, smart watches, etc.) with you.
5. Read the question carefully, understand the question, and then attempt your answers in the provided answer booklet.
6. Verify that you have **eight (08** printed pages of the question paper including this page. There are **six (06)** questions.

Question 1 [9*2 = 18 Marks]

Write a one-line C++ statement for each of the following tasks. The statement must be written clearly and without any cutting/overwriting. If there are two or more ways to perform the given task then choose the best option. The answer must be a single-line statement.

1.	Write a cout statement to display the global variable "Mystery" in the given function: `int Mystery = 10;` `void my_function()` `{` `  int Mystery = 6;` `  //write cout statement to access the Global variable here` `}`
	`std::cout << ::Mystery << std::endl;`
2.	Write a prototype of the function named "**my_func**". The function returns nothing but it takes three parameters x, y and z; where x has a default value of 2, y has no default value and z has a default value of 5.
	`void my_func(int x = 2, int y, int z = 5);`
3.	Declare a pointer **ptr** that can point to any of the two variables holding values 12 and 'a' respectively.

	void* ptr;
4.	Write a statement to reserve enough space for a double-type variable in the stack memory.
	double stack_var;
5.	Write a statement to reserve enough space for a double-type variable in the heap memory.
	double* heap_var = new double;
6.	Assume that an add() function exists (prototype given below). You are asked to make another function which **adds** two characters by concatenating them and returns a string. For example, adding 'a' and 'b' should return "ab". **Just write a prototype** of this required function. int add(int a, int b);
	std::string add(char a, char b);
7.	char *p1 = new char; char *p2 = new char; Write a statement to make p1 dangling pointer.
	delete p1;
8.	Write a statement to undangle pointer p1 after it had become dangling in the previous question.
	p1 = nullptr;
9.	What would be the function prototype (write it) if we could have the following function call? **DO NOT use variable names in the prototype.** char ch = 'Z'; int x=foo(7.5+1, &ch, ch, foo(1.5,&ch,20);
	int foo(double, char*, char, int);

Question 2 [9*2 + 1*6 + 1*5 + 1*10 = 39 Marks]

Write the output of the following C++ codes (if the code is correct). If you find any error/s in the code, please identify, and explain the error/s (Note: do not write output if there is an error). Assume that required libraries and main functions are already included in the program.

10.	**int i = 6, counter = 0;** **while(i-2)** **{** **i++;** **counter++;**

	cout << "Loop is running for " << counter << " time" <<endl; **}**
	Output/Error: **Infinite Loop.**
11.	**#include<iostream.h>** **int MyFunction(int a,int b=3,int c=3)** **{** **cout<< ++a * ++b * --c ;** **return 0;** **}** **int main()** **{** **MyFunction(5, 0, 0);** **return 0;** **}**
	Output/Error: -6
12.	**#include<iostream>** **using namespace std;** **int main()** **{** **int a = 32, *ptr = &a;** **char ch = 'A', *cho = &ch;** **//ASCII of 'A' is 65** ***cho += 3;** ***ptr += ch;** **cout << a << ", " << ch << endl;** **return 0;** **}**
	Output/Error: 100, D
13.	**int my_func(int num)** **{** **cout << num << endl;** **return num*-2;** **}** **int main()** **{** **cout<<my_func(my_func(4)*2);** **return 0;}**
	Output/Error: **4** **-16** **32**

14.	**void f2(int *p){** **int* x = new int;** ***x = 10;** **p = x;** **cout<<*p<<endl;}** **void f1(int *p){** **f2(p);** **}** **int main()** **{** **int x=5;** **int *p=&x;** **f1(p);** **cout<<*p<<endl<<x<<endl;** **return 0;** **}**
	Output/Error: **10** **5** **5**
15.	**int a=20;** **//suppose a is stored at address 0x4B** **int *p=&a;** ***p = 30;** **cout<<a<<endl<<p;**
	Output/Error: **30** **0x4B**
16.	**int x=100,y=200;** **int *const p;** **p = &x;** **cout<<*p<<endl;** **p = &y;** **cout<<*p<<endl;**
	Output/Error: **Compilation error.**
17.	int a = 0, b=36; float f=3.9; b+=(a = 50)*(int)f%3+5.1-6.8*10-b/5; cout << a << "$" << b;
	Output/Error: **50$-33**

18.	char* str1 = "Self-conquest is the greatest victory 😊"; char str2[80]; char* src = str1; char* dest = str2; while(*src) *dest++ = *src++; *dest = '\0'; cout << str2 << endl;
	Output/Error: Self-conquest is the greatest victory 😊
19.	**#include<iostream> using namespace std; int fib(int n=1, int m=2) { static int memo[100] = {0}; if (n <= 1) { return n; } if (memo[n] == 0) { for (int i = 0; i < 2; ++i) memo[n]= (n-1) + (n-2); cout<<memo[n]<<endl; } else cout<<memo[n]<<endl; return memo[n]; } int main() { static int n=3; std::cout << fib(5) << std::endl; std::cout << fib(10) << std::endl; std::cout << fib(5) << std::endl; return 0; }**
[06 Marks]	**Output/Error:** **7 7 17 17 7 7**

20.	``` void magic (int* ptr, int size) { ptr = new int[size+1]; for(int i=size,j=0;i>=0;i--,j++) ptr[j]=i; } int main() { int* ptr = nullptr; int size = 5; magic(ptr,size); for(int i=0; i<size ;i++) cout<<*(ptr + i)<<" "; delete [] ptr; ptr = nullptr; return 0; } ```
[05 Marks]	**Output/Error:** Segmentation fault
21.	``` const char* c[] = { "PF", "Exam" , "centipede", "Project" }; char const * * cp[] = { c + 2, c + 3, c , c + 1 }; char const *** cpp[] = { cp + 1 ,cp + 2 }; int main() { cout << ***cpp[1] << endl; cout << (*cpp)[-1][0] << endl; cout << (*cp)[-1] << endl; cout << (*(cpp[1][-1]) + 3) << endl; return 0; } ```
[10 Marks]	**Output/Error:** P centipede Exam ject

Question 3 [10 Marks]

Write a program to get a number from user and check if that integer is a power of 2. Use bitwise operators to check (see some facts given below for the hint). Input validation is must: number should not be zero, use of any power function is not allowed.

1. If number is power of 2 then then find all roots of a quadratic equation using the nested switch statement. Take a, b, c as input from the user and display the both roots. Format your root values in fixed-point notation, with two decimal places of precision.
2. If it is not power of 2, then display a message that the number is not power of 2.

Note: You may use the sqrt() function of <math.h> library to find the square root.

Power of 2: Some facts about numbers which are power of 2. You may use these facts to check if it is power of 2 or not. Suppose N is power of 2, then:

- all bits of N are zero except (any) one bit.
- All least significant bits of N-1 before the ON bit of N are always ON.

Quadratic equation: In elementary algebra quadratic equation is an equation in the form of

$$ax^2 + bx + c = 0$$

Solving quadratic equation

A quadratic equation can have either one or two distinct real or complex roots depending upon the nature of the discriminant of the equation. Where discriminant of the quadratic equation is given by

$$\Delta = b^2 - 4ac$$

Depending upon the nature of the discriminant, formula for finding roots can be given as:

- Case 1: If the discriminant **is positive**. Then there are two real distinct roots given by. $\frac{-b+\sqrt{\Delta}}{2a}$ and $\frac{-b-\sqrt{\Delta}}{2a}$
- Case 2: If **discriminant is zero**. Then it has exactly one real root given by. $-\frac{b}{2a}$
- Case 3: If **discriminant is negative**. Then it will have two distinct complex roots given by. root 1 = root 2 = $\frac{-b}{2a}$, Imaginary = $\frac{\sqrt{-\Delta}}{2a}$

Solution:

```
#include <iostream>
#include <cmath>

bool isPowerOfTwo(int n) {
   return n && !(n & (n - 1));
}

int main() {
   int num;
   std::cout << "Enter a number: ";
   std::cin >> num;

   if (num == 0) {
      std::cout << "Error: Number should not be zero." << std::endl;
      return 1;
   }
```

```
    if (isPowerOfTwo(num)) {
        double a, b, c;
        std::cout << "Enter coefficients (a, b, c) for quadratic equation (ax^2 + bx + c = 0): ";
        std::cin >> a >> b >> c;

        double discriminant = b * b - 4 * a * c;

        switch (discriminant > 0 ? 1 : (discriminant == 0 ? 2 : 3)) {
            case 1: {
                double root1 = (-b + sqrt(discriminant)) / (2 * a);
                double root2 = (-b - sqrt(discriminant)) / (2 * a);
                std::cout << "Root 1: " << std::fixed << root1 << std::endl;
                std::cout << "Root 2: " << std::fixed << root2 << std::endl;
                break;
            }
            case 2: {
                double root = -b / (2 * a);
                std::cout << "Root: " << std::fixed << root << std::endl;
                break;
            }
            case 3: {
                double realPart = -b / (2 * a);
                double imaginaryPart = sqrt(-discriminant) / (2 * a);
                std::cout << "Root 1: " << std::fixed << realPart << " + " << std::fixed << imaginaryPart <<
"i" << std::endl;
                std::cout << "Root 2: " << std::fixed << realPart << " - " << std::fixed << imaginaryPart <<
"i" << std::endl;
                break;
            }
        }
    } else {
        std::cout << "The number is not a power of 2." << std::endl;
    }

    return 0;
}
```

Question 4 [10 Marks]

Write a function in C++ which takes a number as input and prints the pattern of alphabets as given below. You are required to validate the input that the number should be in-between 1 and 26. For example, for n=5, print pattern as below

```
A B C D E D C B A
  B C D E D C B
    C D E D C
      D E D
        E
```

for n=4, print pattern as below

```
A B C D C B A
```

```
B C D C B
  C D C
    D
```

Solution:

```
#include <iostream>

void printAlphabetPattern(int n) {
    if (n < 1 || n > 26) {
        std::cout << "Error: Number should be between 1 and 26." << std::endl;
        return;
    }

    for (int i = 0; i < n; i++) {
        for (int j = 0; j < i; j++) {
            std::cout << "  "; // Print spaces to align pattern
        }
        char startChar = 'A' + i;
        for (int j = 0; j < n - i; j++) {
            std::cout << startChar++ << " ";
        }
        startChar -= 2;
        for (int j = 0; j < n - i - 1; j++) {
            std::cout << startChar-- << " ";
        }
        std::cout << std::endl;
    }
}

int main() {
    int n;
    std::cout << "Enter a number between 1 and 26: ";
```

```
    std::cin >> n;


    printAlphabetPattern(n);


    return 0;
}
```

Question 5 [10 Marks]

Write a C++ program to input an upper limit and lower limit and check whether the numbers in that range are Strong or not. The upper limit should be always greater than the lower limit, upper and lower should be greater or equal to zero. Write the program neatly with proper variable names, comments and indentation.

What is a Strong number?
A strong number is a special number whose sum of factorial digits is equal to the original number. For example: 145 is a strong number. Since, 1! + 4! + 5! = 145

Program design	Marks
Proper Input and its validation	2
Using functions (proper call, definition, prototype)	3
Program Logic	4
Correct use of loops	3
Not using comments and indentation	-2

Solution:

```
#include <iostream>

// Function to calculate the factorial of a number
int factorial(int n) {
    if (n == 0 || n == 1) {
        return 1;
    } else {
        return n * factorial(n - 1);
    }
}

// Function to check if a number is strong or not
bool isStrong(int num) {
    int originalNum = num;
    int sum = 0;

    while (num > 0) {
        int digit = num % 10;
        sum += factorial(digit);
        num /= 10;
    }

    return sum == originalNum;
}
```

```
int main() {
   int lowerLimit, upperLimit;

   // Input lower limit
   do {
      std::cout << "Enter lower limit (greater than or equal to 0): ";
      std::cin >> lowerLimit;
   } while (lowerLimit < 0);

   // Input upper limit
   do {
      std::cout << "Enter upper limit (greater than lower limit and greater than or equal to 0): ";
      std::cin >> upperLimit;
   } while (upperLimit < lowerLimit || upperLimit < 0);

   std::cout << "Strong numbers between " << lowerLimit << " and " << upperLimit << " are: ";

   // Check and print strong numbers within the range
   for (int i = lowerLimit; i <= upperLimit; ++i) {
      if (isStrong(i)) {
         std::cout << i << " ";
      }
   }

   std::cout << std::endl;

   return 0;
}
```

Question 6 [15 Marks]

Write a function that accepts an integer 2D array of size 6x6 to find pairs and indexes of elements that have the maximum and minimum difference among all element pairs.

Example input:

1	**2**	**5**	**7**	**9**	**11**
13	**15**	**17**	**19**	**78**	**21**
23	**36**	**38**	**78**	**40**	**42**
44	**46**	**48**	**50**	**52**	**54**
56	**58**	**60**	**62**	**100**	**64**
66	**68**	**70**	**72**	**74**	**76**

Output : Minimum distance between **1** and **2,** at index **(0,0)** and **(0,1)**
Maximum distance between **1** and **100,** at index**(0,0)** and **(4,4)**

Solution:

```
#include <iostream>
#include <cmath>

void findMinMaxDifference(int arr[][6]) {
    int minDiff = INT_MAX, maxDiff = INT_MIN;
    int minIndexI, minIndexJ, maxIndexI, maxIndexJ;

    // Iterate through the array to find minimum and maximum differences
    for (int i = 0; i < 6; ++i) {
        for (int j = i + 1; j < 6; ++j) {
            int diff = abs(arr[i][j] - arr[j][i]);
            if (diff < minDiff) {
                minDiff = diff;
                minIndexI = i;
                minIndexJ = j;
            }
            if (diff > maxDiff) {
                maxDiff = diff;
                maxIndexI = i;
                maxIndexJ = j;
            }
        }
    }

    // Output the results
    std::cout << "Minimum distance between " << arr[minIndexI][minIndexJ] << " and " <<
arr[minIndexJ][minIndexI] << ", at index (" << minIndexI << "," << minIndexJ << ") and (" <<
minIndexJ << "," << minIndexI << ")" << std::endl;
    std::cout << "Maximum distance between " << arr[maxIndexI][maxIndexJ] << " and " <<
arr[maxIndexJ][maxIndexI] << ", at index (" << maxIndexI << "," << maxIndexJ << ") and (" <<
maxIndexJ << "," << maxIndexI << ")" << std::endl;
}

int main() {
    int arr[6][6] = {
        {1,  2,  5,  7,   9,  11},
        {13, 15, 17, 19,  78, 21},
        {23, 36, 38, 78,  40, 42},
        {44, 46, 48, 50,  52, 54},
        {56, 58, 60, 62, 100, 64},
        {66, 68, 70, 72,  74, 76}
    };

    findMinMaxDifference(arr);

    return 0;
}
```