DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

I SEMESTER B.TECH.

Mid Term Exam

SUBJECT: Programming for Problem Solving [CSE 1171]

Time: 8:30 AM Date: 27-09-2024 MAX.MARKS: 30

ANSWER SCHEME

Q.No	Questions	Marks	CO	AHEP LO	BT
1.	Outline the role of the return 0; statement in the main function 1. It defines a function **2. It returns a value indicating successful program execution** 3. It terminates the program abruptly 4. It starts a loop	**0.5**	**1**		**2**
2.	Which of the following types of memory is volatile? a) ROM b) **Cache memory** c) Hard disk d) Optical storage	**0.5**	**1**		**1**

3.	What will be the output of the following error free C code?
	#include <stdio.h>
	int main()
	{
	int i, j;
	for (i = 2; i < 10; i++) {
	for (j = 2; j <= (i/j); j++)
	if (!(i % j))
	break;
	if (j > (i/j))
	printf("%d ", i);
		0.5	**2**		**5**

	}return 0; } 1. 2 3 4 5 6 7 8 9 2. 3 5 7 9 **3. 2 3 5 7** 4. 2 3 5 7 11				

4.	What is the output of this C code?
	#include <stdio.h>
	int main()
	{
	int i = 0, j = 0;
	while (i<5&&j<10)
	{
	i++;
	j++;
	}
	printf("%d,%d\n",i,j);
	return 0; }
	1. 5, 5
	2. 10,10
	3. 5,10
	4. error
		0.5	**2**		**5**

5.	The expression (5>1 || 6<1) evaluates to: (0.5) **1. 1** 2. 0 3. 2 4. -1	**0.5**	**2**		**2**

6.	In the following switch statement, what will be the output if the user enters 5 as the value of x?
	switch(x % 3) {
	case 0:
	printf("Zero");
	break;
	case 1:
	printf("One");
	case 2:
	printf("Two");
	break;
	default:
	printf("Default");
	}
	a) Two
	b) One
	c) Zero
	d) OneTwo
	Answer: c) Two
		0.5	**2**		**3**

7.	What will be the value of sum after the execution of the following do-while loop?
	int i = 5, sum = 0;
	do {
	sum += i;
	i--;
	} while(i > 1);
	a) 10
	b) 14
	c) 15
	d) 20
	Answer: b) 14
		0.5	**2**		**3**

8.	How many times will the following for loop iterate?
	for(int i = 1; i < 20; i *= 2) {
	printf("%d ", i);
	}
	a) 4 times
	b) 5 times
	c) 6 times
	d) 20 times
	Answer: b) 5 times
		0.5	**2**		**3**

9.	Identify the output of the following code?
	int arr[] = {2, 4, 6, 8, 10};
	int i;
	for (i = 0; i < 5; i++) {
	if (arr[i] % 4 == 0) {
	printf("%d", arr[i]);
	break;
	}
	}
	a) 2
	b) 4
	c) 6
	d) 8
		0.5	**3**		**3**

10.	Identify the output of the following program
	#include<stdio.h>
	int main()
	{
	int arr[4][5], i,j;
	for(i=0;i<4;i++)
	{
	for(j=0;j<5;j++)
	{
	arr[i][j]=10*i+j;
	}
	}
	printf("%d",arr[1][1]+9);
	return 0;
	}
	a) 10
	b) 15
		0.5	**3**		**2**

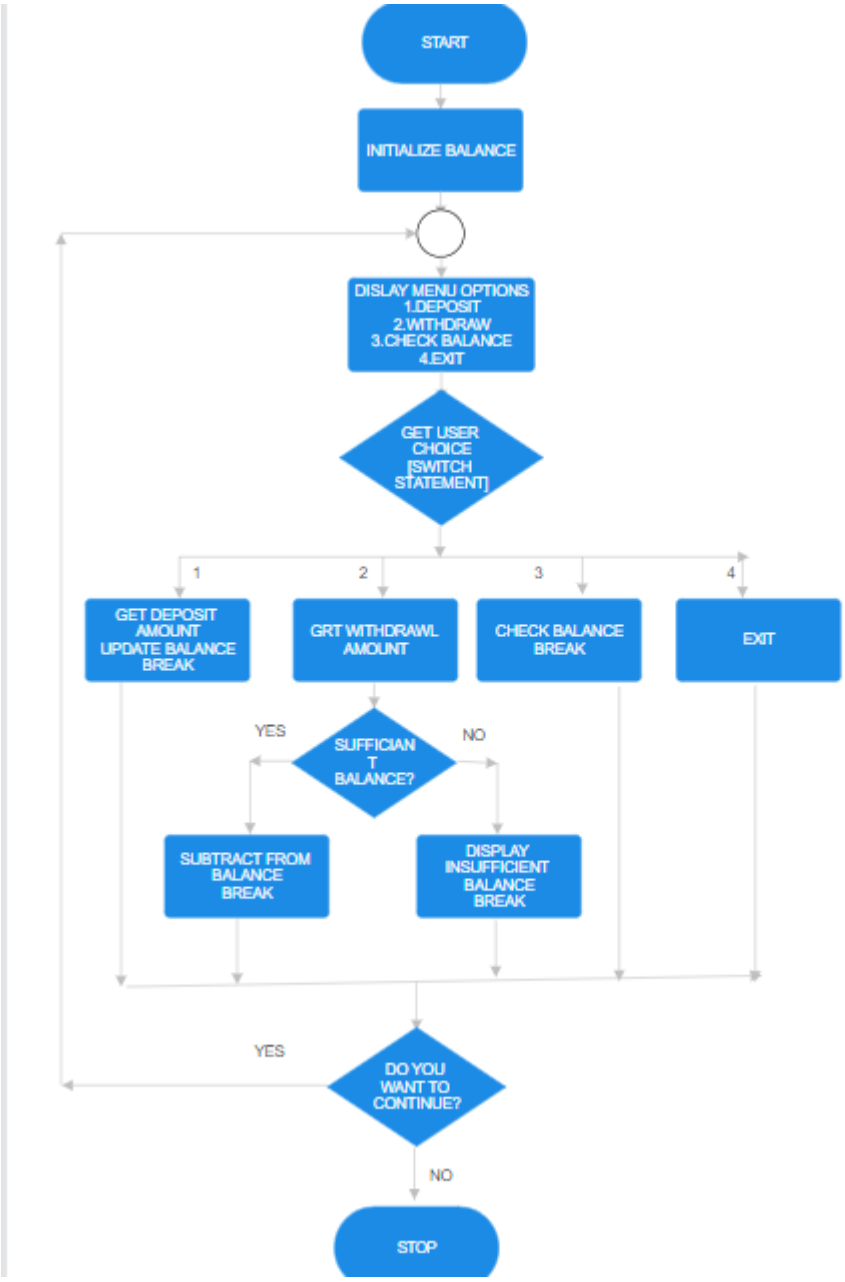
	c) 20 d) 29				

11.	Create a C program that generates a prime factorization pattern based on user input. The program should:
	1. Ask the user to input a positive integer n (4 <= n <= 50).
	2. For each non-prime numbers from 2 to n, display it's all factors.
	3. For prime numbers between the limits, display "PRIME".
	Example: Enter a number (4 <= n <= 50): 10
	2: PRIME
	3: PRIME
	4: Factors: 1 2 4
	5: PRIME
	6: Factors: 1 2 3 6
	7: PRIME
	8: Factors: 1 2 4 8
	9: Factors: 1 3 9
	10: Factors: 1 2 5 10
	
	ANSWER
	#include <stdio.h>
	
	int main() {
	
	int n, i, j, isPrime;
	
	// Ask the user to input a number within the range 4 <= n <= 50 ---------- (1 M)
	
	do {
	
	printf("Enter a number (4 <= n <= 50): ");
	
	scanf("%d", &n);
	
	} while (n < 4 || n > 50);
	
	// Loop through each number from 2 to n ---------- (1M)
	
	for (i = 2; i <= n; i++) {
	
	isPrime = 1; // Assume the number is prime initially
		4M	**2**		**3**

```
    // Check if the number is prime by trying to divide it by numbers < i----------(1
M)
    for (j = 2; j  < i; j++) {
      if (i % j == 0) {
        isPrime = 0; // If divisible, it's not prime
        break;
      }
    }

    if (isPrime) {
      // If the number is prime, print "PRIME"
      printf("%d: PRIME\n", i);
    } else {
      // If not prime, print its factors----------------------------------------------(1M)
      printf("%d: Factors: ", i);
      for (j = 1; j <= i; j++) {
        if (i % j == 0) {
          printf("%d ", j); // Print each factor of i
        }
      }
      printf("\n");
    }
  }

  return 0;
}
```

12.	Design a flowchart that simulates a basic banking system. The flowchart should display a menu of options: **Deposit**, **Withdraw**, **Check Balance**, and **Exit**. Use a switch statement to handle user input and update the account balance accordingly.  Any appropriate flowchart which defines well as per the question will be considered accordingly	**3M**	**1**		**6**
13.	Assuming a=8 ,b=15 and c=4, evaluate the following expression: 2*((a%5)*(4+(b-3)/(c+2))) [Steps for expression evaluation-3M,only answer without steps-0.5M] 2*((8%5)*(4+(15-3)/(4+2)))	**3**	**2**		**3**

	2*(3*(4+(15-3)/(4+2))) 2*(3*(4+12/(4+2))) 2*(3*(4+12/6)) 2*(3*(4+2)) 2*(3*6) 2*18=36				

14.	Differentiate between implicit and explicit type casting with suitable examples. [differences-2M, Example-1M]
	
	In C programming, type casting is used to convert a variable from one data type to another. There are two main types of type casting: implicit and explicit.
	
	Implicit Type Casting
	
	Implicit type casting (also known as automatic type conversion) is performed by the compiler automatically. It usually occurs when you mix different data types in an expression. The compiler promotes the lower type to the higher type to ensure that the operation is performed correctly.
	
	Example:
	
	#include <stdio.h>
	
	int main() {
	  int a = 10;
	  float b = 5.5;
	  float result;
	
	  result = a + b; // Implicit type casting
	
	  printf("Result: %f\n", result); // Output will be 15.5
	  return 0;
	}
	
	In this example:
	
	• a is an int and b is a float.
	• When a and b are added together, a is implicitly cast to a float so that the addition is performed using float arithmetic.
	• The result is a float, and this value is assigned to the result variable, which is also of type float.
		3	**2**		**2**

	Explicit Type Casting
	
	Explicit type casting is performed manually by the programmer using type casting operators. This allows for more control over the conversion process and can be used to convert between types where implicit casting may not occur or may lead to loss of data.
	
	Example:
	
	#include <stdio.h>
	
	int main() {
	  float x = 9.75;
	  int y;
	
	  y = (int)x; // Explicit type casting
	
	  printf("Value of y: %d\n", y); // Output will be 9
	  return 0;
	}
	
	In this example:
	
	• x is a float and y is an int.
	• (int)x explicitly casts the float value of x to an int. This truncates the decimal part and only the integer part is stored in y.
	• The output shows the integer value 9, which results from the explicit type conversion.				

15	Develop an algorithm to find the greatest common divisor (GCD) of two positive integers.	**3**	**1**		**3**

	Name of the Algorithm: GCD of 2 positive integers Step 1: Start Step 2: Input a, b Step 3: if (b==0) return a as GCD Goto Step 7 Step 4: remainder ←a%b Step 5: a ← b Step 6: b ←remainder Goto Step 3 Step 7: Print GCD Step 8: Stop				
16	Illustrate the working of binary search for the given 1D array [3, 6, 9, 12, 15, 18, 23, 30, 35, 40] to search the number 23. (Note: Do not write algorithm, flowchart or program.) **Scheme:** **Illustration:** In each steps all index values shown give full 3M · **Step 1**: • low = 0, high = 9 • Calculate mid: mid=0+9−02=4 • The middle element at index 4 is 15. • Since 23 > 15, search the right half by setting low = 5. · **Step 2**: • low = 5, high = 9 • Calculate mid: mid=5+9/2=7 • The middle element at index 7 is 30. • Since 23 < 30, search the left half by setting high = 6. · **Step 3**: • low = 5, high = 6 • Calculate mid: mid=5+6/2=5 • The middle element at index 5 is 18. • Since 23 > 18, search the right half by setting low = 6. · **Step 4**: • low = 6, high = 6 • Calculate mid: mid=6+6/2=	**3**	**3**		**2**

	• The middle element at index 6 is 23, which is the target value.				

17	Compare and contrast ‘break’ and ‘continue’ statements in C programming. Provide a simple example for each to illustrate their usage.
	Answer:
	The main differences between 'break' and 'continue' are:
	
	break: Terminates the loop entirely and exits to the first statement after the loop.
	continue: Skips the rest of the current iteration and moves to the next iteration of the loop.
	
	break: Stops the loop from executing further iterations.
	continue: Allows the loop to continue with the next iteration.
	
	Examples:
	
	<u>break example:</u>
	
	`for (int i = 1; i <= 5; i++) {`
	`  if (i == 3) {`
	`  break;`
	`  }`
	`  printf("%d ", i);`
	`}`
	`// Output: 1 2`
	
	<u>continue example:</u>
	
	`for (int i = 1; i <= 5; i++) {`
	`  if (i == 3) {`
	`  continue;`
	`  }`
	`  printf("%d ", i);`
	`}`
	`// Output: 1 2 4 5`
	Mark Distribution:
	
	Correct explanation of the difference between break and continue: 1 mark
	Providing appropriate examples for both break and continue: 1 mark
		2	**3**		**4**

18	Write a C program to read MxM 2D array and reverse the elements of each row of the matirx. Display the original and modified array in matrix format in main. Example:	**2M**	**3**		**2**

Input:
Original matrix elements
1 2 3
4 5 6
7 8 9
Output:
Modified matrix elements
3 2 1
6 5 4
9 8 7

Scheme: Input/Output 1M, Logic 1M

```
int main() {
   int M;

   // Read the size of the array
   printf("Enter the size of the 2D array (MxM): ");
   scanf("%d", &M);

   int arr[100][100];  // Assuming a maximum size of 100x100

   // Input the elements of the array
printf("Enter the elements of the %dx%d array:\n", M, M);
   for (int i = 0; i < M; i++) {
      for (int j = 0; j < M; j++) {
         scanf("%d", &arr[i][j]);
      }
   }

   // Display the original array
   printf("\nOriginal Array:\n");
 for (int i = 0; i < M; i++) {
      for (int j = 0; j < M; j++) {
         printf("%d ", arr[i][j]);
      }
      printf("\n");
   }

   // Reverse each row of the array
  for (int i = 0; i < M; i++) {
      int start = 0;
      int end = M - 1;
      while (start < end) {
         // Swap elements at start and end
         int temp = arr[i][start];
         arr[i][start] = arr[i][end];
         arr[i][end] = temp;
         start++;
         end--;
```

```
        }
    }
    // Display the modified array
    printf("\nModified Array (with reversed rows):\n");
    for (int i = 0; i < M; i++) {
        for (int j = 0; j < M; j++) {
            printf("%d ", arr[i][j]);
        }
        printf("\n");
    }
    return 0;
}
```

OR

```
#include <stdio.h>


int main() {


  int i,j, a[10][10],m,n,temp;
  printf("enter the no. of rows and columns:");
  scanf("%d%d",&m,&n);
  printf("enter the elements of matrix\n");
  for(i=0;i<m;i++)
  {
     for(j=0;j<n;j++)
     scanf("%d",&a[i][j]);
  }
  for(i=0;i<m;i++)
  {
     for(j=0;j<n;j++)
     printf("%d\t",a[i][j]);
     printf("\n");
  }
  for(i=0;i<m;i++)
```

	{
	for(j=0;j<n/2;j++)
	{
	temp=a[i][j];
	a[i][j]=a[i][n-j-1];
	a[i][n-j-1]=temp;
	}
	}
	printf("the resultant matrix is:\n\n");
	for(i=0;i<m;i++)
	{
	for(j=0;j<n;j++)
	printf("%d\t",a[i][j]);
	printf("\n");
	}
	
	return 0;
	}				

19	Write a c program to sort a given input array using bubble sort. **Answer:** **Scheme: Input/Output 1M, Logic 1M** #include <stdio.h> #define MAX_SIZE 100 int main() { int arr[MAX_SIZE]; int n, i, j, temp;	**2M**	**2**		**3**

```
    // Input array size
    printf("Enter the number of elements (max %d): ", MAX_SIZE);
    scanf("%d", &n);

    if (n <= 0 || n > MAX_SIZE) {
        printf("Invalid array size. Please enter a number between 1 and %d.\n",
MAX_SIZE);
        return 1;
    }

    // Input array elements
    printf("Enter %d integers:\n", n);
    for (i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }

    // Print original array
    printf("Original array: ");
    for (i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");

    // Bubble sort
    for (i = 0; i < n-1; i++) {
        for (j = 0; j < n-i-1; j++) {
            if (arr[j] > arr[j+1]) {
                // Swap elements
```

```
            temp = arr[j];
            arr[j] = arr[j+1];
            arr[j+1] = temp;
        }
    }
}

// Print sorted array
printf("Sorted array: ");
for (i = 0; i < n; i++) {
    printf("%d ", arr[i]);
}
printf("\n");

return 0;
}
```