Dept. of Computer Science & Engineering
Course: Problem Soving Using Computers (PSUC), CSE 1071
First Semester BTech (Core Branches) Midterm Exam – Sept. 2024
Solution and Scheme of Evaluation

MCQ
Answer all the questions.

Q1. What is the output of the following C program?

```
#include <stdio.h>
int main()
    {
      int a=5, b=7;
      float n=3.5, m=2.1;
      printf("a+b=%d\tn-m=%.3f", a+b, n-m);
      return 0;
    }
```

(1)

1. a+b=12.000
   n-m=1.4
2. a+b=12.000
   n-m=1.400
3. **a+b=12**
   n-m=1.400
4. a+b=12 n-m=1.400

Q2. What is the output of the following C program?

```
#include <stdio.h>
int main()  {
 int y, a=2, b=20, c=4;
 y = a*(3/(4-b/(c*2)));
 printf("%d",y);
 return 0;    }
```

(1)

1. 0
2. 1
3. **2**
4. -2

Q3. What is the output of the following C program?

```
#include <stdio.h>
int main()  {
   float x = 4.6;
   double y = 4.6;
   if(x == y)
          printf("Equal");
   else if(x > y)
```

```
        printf("x > y");
    else if(x < y)
        printf("x < y");
    else
        printf("x not equal to y");
    return 0;    }
```
(1)

1. Equal
2. **x < y**
3. x > y
4. x not equal to y

Q4. What is the output of the following C program?

```
#include <stdio.h>
int main()  {
    int x=15;
    int count = 0;
    while (x > 0)  {
        x = (x & (x - 1));
        count++;
       }
    printf("%d", count);
    return 0;  }
```
(1)

1. 1
2. 2
3. 3
4. **4**

Q5. What is the output of the following C program?

```
#include <stdio.h>
int main()  {
    int arr[ ] = {1, 2, 3, 4, 5, 6};
    int i, sum = 0;
    int size = sizeof(arr) / sizeof(arr[0]);
    for (i = 0; i < size; i += 2)
        sum += arr[i];
    printf("%d", sum);
    return 0;    }
```
(1)

1. 18
2. 12
3. **9**
4. 16

DESCRIPTIVE

Answer all the questions.

Q6. Differentiate among machine level, assembly level and high level languages. (3)

Solution:

Machine-level, assembly-level, and high-level languages differ primarily in abstraction, ease of use, and proximity to the hardware:

1. Machine-Level Language (Machine Code)

- **Definition**: This is the lowest level of programming language, directly understood by the computer's hardware (CPU).
- **Nature**: Binary (0s and 1s), which the machine's processor can execute directly.
- **Example**: 01001011 01101000 11100001 (Specific binary codes for operations).
- **Pros**:
  - Executes extremely fast.
  - Direct control over hardware.
- **Cons**:
  - Extremely difficult to write, debug, and maintain.
  - Not portable—machine code is specific to a given CPU architecture.

2. Assembly-Level Language (Assembly Code)

- **Definition**: A low-level language that uses symbolic representations of machine instructions.
- **Nature**: Uses mnemonics (shortcodes) and requires an assembler to convert the code to machine language.
- **Example**: MOV AX, 01h (Move the hexadecimal value 1 into the AX register).
- **Pros**:
  - More readable than machine code.
  - Still allows for fine-grained control over hardware.
- **Cons**:
  - Harder to learn and use than high-level languages.
  - Not portable—assembly code is specific to a particular CPU architecture.

3. High-Level Language

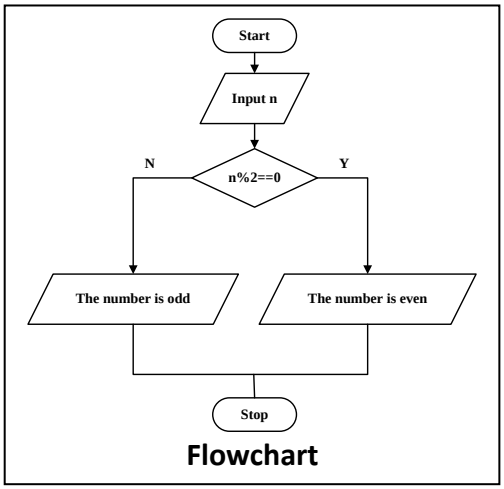
- **Definition**: A programming language that is more abstracted from the hardware and designed to be easy for humans to understand and use.
- **Nature**: Uses English-like syntax and concepts, and is translated to machine code by a compiler or interpreter.
- **Example**: printf("Hello, World!") (C language).
- **Pros**:
  - Easier to learn, write, and debug.
  - Portable across different systems (compilers or interpreters handle architecture differences).
- **Cons**:
  - Less control over hardware.
  - May be slower compared to assembly and machine code due to higher abstraction levels.

Scheme : Machine level language: 1M, Assembly level language: 1M & High level languages: 1M

Q7.Write an algorithm and flowchart to find whether the number is even or odd. (3)

Algorithm:

Step 1: Start
Step 2: Input a number 'n'
Step 3: Check the remainder by using n%2
Step 4: If the remainder is zero, go to step 5, else go to Step 6.
Step 5: Print 'The number is even'. Go to Step 7.
Step 6: Print ' The number is odd'.
Step 7: Stop



Flowchart

Scheme : Algorithm: 1.5 M, Flowchart: 1.5 M

Q8. What is a ternary operator in C? With proper syntax, explain the ternary (conditional) operator. Write a complete C program to check whether the given year is a leap year using the ternary operator. (4)

Solution:

As the name indicates, an operator that operates on three operands is called a ternary operator. C has a conditional or ternary operator pair (? :) that uses a boolean condition to determine which of two expressions is evaluated. **[1 Mark]**

Syntax:

condition ? expression1 : expression2

condition can be true or false.

If the condition is **true**, expression1 is evaluated; if it is **false**, expression2 is evaluated.

Example:

max=(a>b) ? a : b; // Compute maximum of a and b **[1 Mark]**

Program:

```
// C program to check leap year using conditional (ternary) operator.
#include <stdio.h>
int main()  {
   int year;
   printf("Enter any year: ");
   scanf("%d", &year);
   (year%4==0 && year%100!=0) ? printf("LEAP YEAR") : (year%400 ==0 ) ? printf("LEAP  YEAR") :
                                                                  printf("COMMON YEAR");
   return 0;
 }
```

[Scheme: Input & output: 1 Mark, correct use of ternary operator: 1 Mark]

Q9.Write a C program to calculate income tax based on the tax slabs for individuals. The program should prompt the user to input their annual income and display the total tax payable according to the following tax slabs. The income tax calculation is as follows:

1. For income up to Rs. 2,50,000: No tax
2. For income Rs. 2,50,001 to Rs. 5,00,000: 5% on the income exceeding Rs. 2,50,000
3. For income Rs. 5,00,001 to Rs. 10,00,000: 20% on the income exceeding Rs. 5,00,000
4. For income above Rs. 10,00,000: 30% on the income exceeding Rs. 10,00,000
Also, include an additional 4% health and education cess on the calculated tax.
Expected sample output:
Enter your annual income: Rs. 7,50,000
Income Tax: Rs. 62,500
Health and Education Cess: Rs. 2,500
Total Tax Payable: Rs. 65,000 **(3)**

Solution:

```
#include <stdio.h>
int main() {
   float income, tax = 0.0, cess = 0.0, totalTax;

   // Reading the annual income
   printf("Enter your annual income (in rupees): ");
   scanf("%f", &income);

   // Income tax calculation based on slabs
   if (income <= 250000) tax = 0;  // No tax for income up to ₹2,50,000

   // 5% tax on income between ₹2,50,001 and ₹5,00,000
   else if (income <= 500000) tax = (income - 250000) * 0.05;

   // 20% tax on income between ₹5,00,001 and ₹10,00,000
   else if (income <= 1000000) tax = (250000 * 0.05) + ((income - 500000) * 0.2);

   // 30% tax on income above ₹10,00,000
   else tax = (250000 * 0.05) + (500000 * 0.2) + ((income - 1000000) * 0.3);

   // Adding 4% health and education cess on tax
   cess = tax * 0.04;
   totalTax = tax + cess;

   // Displaying the results
   printf("\nIncome Tax: Rs.%.2f", tax);
   printf("\nHealth and Education Cess: Rs.%.2f", cess);
   printf("\nTotal Tax Payable: Rs.%.2f\n", totalTax);

   return 0;
}
```

[Scheme: input: 0.5M, output: 0.5M, Income tax calculation: 2M]

Q10. (A) It is required to write a C program to guess a secret number that is defined as an integer constant within the C program. (Let the secret number be 5).
Conditions to be satisfied by the program:

- **Let the secret number take values between 1 and 10.**
- **The program should use a loop to continuously allow the user to enter a guess number.**
- **If the guess number is greater than the secret number, it should display 'Greater than secret number'.**
- **If the guess number is lesser than the secret number, it should display 'Lesser than secret number'.**
- **If the guess number is equal to the secret number, it should display 'Congratulations, You have guessed it right'**
- **The program should exit only when the user enters a value of -1.**

Suggest which loop construct of C would you select to solve the above problem? Justify your selection.(DO NOT write the program or algorithm or flowchart)
(B) With the help of a suitable example, explain the exit-controlled loop. (3)
[Solution & Scheme:

(A) Loop to be selected: Do-While loop **[0.5M]**; Justification: The body of loop should be executed atleat once **[0.5M]**
(B) Explanation of exit-controlled loop (Do-While Loop): (Syntax and flow chart expected): **[1M]**
Example C program on Do-While Loop: **[1M]**

Q11. Write a C program to:
(i) read the elements of a 1D array 'arr' from the user
(ii) copy all even numbers from the array 'arr' to another 1D array named 'even' and all odd numbers from array 'arr' to a third 1D array named 'odd', and
(iii) print all the resulting arrays. (4)

```
#include <stdio.h>
int main()
 {
   int n, arr[20], even[10], odd[10],i, neven=0, nodd=0;

   //Reading no. of elements in arr
   printf("Enter no. of elements in arr:\n");
   scanf("%d",&n);

   //Reading elements of arr
   printf("Enter elements of arr:\n");
   for(i=0; i<n; i++)
       scanf("%d",&arr[i]);

   //Identify even & odd nos. and store in separate arrays
   for(i=0; i<n; i++)
      {
        if(arr[i]%2==0)        even[neven++]=arr[i];
        else                   odd[nodd++]=arr[i];
      }
```

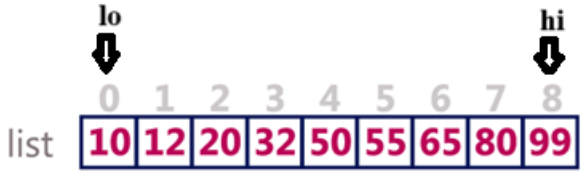
```
    //Displaying elements of main array
    for(i=0; i<n; i++)
        printf("arr[%d] = %d\t",i,arr[i]);
    printf("\n");

    //Displaying elements of even array
    for(i=0; i<neven; i++)
        printf("even[%d] = %d\t",i,even[i]);
    printf("\n");

    //Displaying elements of odd array
    for(i=0; i<nodd; i++)
        printf("odd[%d] = %d\t",i,odd[i]);

    return 0;
}
```

[Scheme: Reading array: 1M, Separating even and odd nos. in separate arrays: 2M, Display elements of even nos. array: 0.5 M, Display elements of odd nos. array: 0.5 M.]

Q12. Illustrate the binary search technique to search 80 in a given 1D array 10, 12, 20, 32, 50, 55, 65, 80, 99. Show clearly all the required intermediate steps. (Note: Do not write algorithm or program). (2)

Solution:

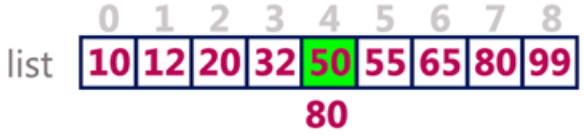
Illustration:



search element **80**

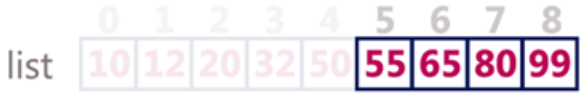
Step 1: **mid = lo + hi / 2 = 0 + 8 / 2 = 4**

search element (80) is compared with middle element (50)



Both are not matching. And 80 is larger than 50. So we search only in the right sublist (i.e. 55, 65, 80 & 99).

So, new **lo = mid + 1 = 4 + 1 = 5**

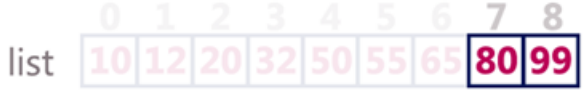


Step 2: **mid = 5 + 8 / 2 = 6**

search element (80) is compared with middle element (65)

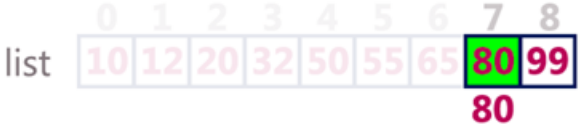
Both are not matching. And 80 is larger than 65. So we search only in the right sublist (i.e. 80 & 99).

So, new **lo = mid + 1 = 6 + 1 = 7**



Step 3: **mid = 7 + 8 / 2 = 7**

search element (80) is compared with middle element (80)



Both are matching. So the result is "Element found at index 7"

[Scheme: Overall Illustration: ½ Marks; each correct step carries: ½ Marks]

Q13. Imagine you are developing a system for a car service station that needs to manage the car records. Each car record has a unique ID number. Due to an emergency, the system must quickly sort these records by the car ID to prioritize the cars with the lowest IDs first. Write a C program that sorts an array of car IDs using Selection Sort algorithm. (3)

Solution:

```
#include <stdio.h>
int main()
{
   int CarIDs[100], n, i, j, minIndex, temp, newID, pos;

   // Input the number of cars
   printf("Enter the number of cars: ");
   scanf("%d", &n);

   // Input the Car IDs
   printf("Enter the Car IDs:\n");
   for (i = 0; i < n; i++)    scanf("%d", &CarIDs[i]);

   // Sort the Car IDs using Selection Sort
   for (i = 0; i < n-1; i++)    {   // Finding the minimum element in the unsorted part of the array
       minIndex = i;
       for (j = i+1; j < n; j++)
          if (CarIDs [j] < CarIDs[minIndex])  minIndex = j;

       // Swap the found minimum element with the first element
       temp = CarIDs [minIndex];
       CarIDs[minIndex] = CarIDs[i];
       CarIDs[i] = temp;
      }
   // Display the car IDs after sorting
   printf("\nCar IDs after sorting: ");
   for (int i = 0; i < n; i++)       printf("%d ", CarIDs[i]);
   printf("\n");

   return 0;
}
```

[Marking Scheme: input: 1M; Selection Sort: 1M and output: 1M]

#####