A REPORT ON PROGRAMMING ASSIGNMENTS USING C

MD. ANAMUL HAQUE | ID: 232000120e | CSE 104
SUMMER 2023 | AUGUST 14, 2023

Problem:

1. Write logical expressions that tests whether a given character variable ***c*** is
    - lower case letter
    - upper case letter
    - digit
    - white space (includes space, tab, new line)

Code :

```
#include <stdio.h>
#include <ctype.h>
int main() {
    char c;

    printf("Enter a character: ");
    scanf("%c", &c);

    if (isupper(c)) {
        printf("The character is an uppercase letter\n");
    } else if (islower(c)) {
        printf("The character is a lowercase letter\n");
    } else if (isdigit(c)) {
        printf("The character is a digit\n");
    } else if (isspace(c)) {
        printf("The character is a white space
character\n");
    } else {
        printf("The character is not uppercase,
lowercase, digit, or white space\n");
    }

    return 0;
}
```

Explanation :

The program includes <ctype.h> (for character handling functions).

A variable named c of type char is declared. This variable will store the character entered by the user.

The printf() function is used to display a prompt to the user, asking them to enter a character.

The scanf() function is used to read a character from the user and store it in the variable c.

The program uses if-else-if statements to determine the type of the character entered by the user.

The isupper() library function is used to check if the entered character is an uppercase letter. If it is true, the program prints a message the character is an uppercase letter.

If the character is not an uppercase letter, the program uses the islower() function to check if it's a lowercase letter. If it is true, the program prints a message the character is a lowercase letter.

If the character is neither an uppercase letter nor a lowercase letter, the program uses the isdigit() function to check if it's a digit. If it is true, the program prints a message the character is a digit.

If the character is not an uppercase letter, lowercase letter, or digit, the program uses the isspace() function to check if it's a white space character (includes space, tab, new line) . If it is, the program prints a message indicating that the character is a white space character.

If the character doesn't fall into any of the above categories, it prints a message The character is not uppercase, lowercase, digit, or white space.

Problem:

2. Write a program to print the numbers between 1 and 10, along with an indication of whether each is even or odd, like this:

```
1 is odd
2 is even
3 is odd
```

Code :

```
#include <stdio.h>
int main() {
   int i;
   for(i=1;i<=10;i++){
    if(i%2==0)
        printf("%d is Even\n",i);
    else
        printf("%d is odd\n",i);
   }

    return 0;
}
```

Explanation :

int I : Declares an integer variable named i, which will be used as a loop counter.

for(i=1; i<=10; i++): This is a for loop that initializes i to 1, executes the loop as long as i is less than or equal to 10, and increments i by 1 after each iteration.

Inside the loop, the program uses the modulus operator (%) to check whether i is even or odd. If i % 2 equals 0, it means that i is divisible by 2 and thus is even. If the condition is not met (i.e., i % 2 is not 0), then i is odd.

Depending on whether i is even or odd, the program uses the printf() function to print a message to the console. If i is even, it prints %d is Even, where %d is a placeholder for the value of i. Otherwise, it prints %d is odd, again with %d being replaced by the value of i.

Problem:

3. Write a ***square()*** function and use it to print the squares of the numbers 1-10:

```
1  1
2  4
3  9
4  16
...
9  81
10 100
```

Code :

```
#include<stdio.h>
int main(){

    int sq,i;

    for(i=1;i<=10;i++){
        sq = square(i);
        printf("\nSquare of %d is = %d ",i,sq);
    }
    return 0;
}
int square(int num){
    return num*num;
}
```

Explanation :

int sq, i;: Declares integer variables sq (for the square) and i (loop counter).

for (i = 1; i <= 10; i++): This is a for loop that iterates from 1 to 10 (inclusive).

Inside the loop, the program calls the square() function, passing the value of i as an argument. The square() function calculates the square of the input number.

printf("\nSquare of %d is = %d ", i, sq);: This line prints the square of the current number i along with the corresponding square value sq.

int square(int num): This is the definition of a separate function named square, which takes an integer input num and returns the square of that number.

Problem:

4. Write a function to compute the factorial of a number, and use it to print the factorials of the numbers 1-7.

Code :

```
#include <stdio.h>
int main()
{
    int i;
    for (i = 1; i <= 7; i++)
    {
    printf("\nfactorial of %d is = %d ", i,
    factorial(i));
    }
    return 0;
}
int factorial(int num)
{
    int j, fact = 1;
    for (j = 1; j <= num; j++)
    {
        fact *= j;
    }
    return fact;
}
```

Explanation :

int i; Declares an integer variable i for the loop counter.

for (i = 1; i <= 7; i++): This is a for loop that iterates from 1 to 7

Inside the loop, the program calls the factorial() function, passing the value of i as an argument. The factorial() function calculates the factorial of the input number.

printf("\nfactorial of %d is = %d ", i, factorial(i));: This line prints the factorial of the current number i along with the calculated factorial value.

int factorial(int num): This is the definition of a separate function named factorial, which takes an integer input num and calculates the factorial of that number using a loop.

int j, fact = 1;: Declares integer variables j for the loop counter and fact to store the factorial value.

for (j = 1; j <= num; j++): This is a for loop that iterates from 1 to num to calculate the factorial.

fact *= j;: Multiplies the current value of fact by the loop counter j, updating the factorial value in each iteration.
return fact;: Returns the calculated factorial value to the caller.

Problem:

5. Write a C program that calculates the HCF and LCM of two numbers.

Code :

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

int main() {
    int num1 , num2 , i ,hcf ,lcm ;
    printf("Enter two positive integer : ");
    scanf("%d %d",&num1,&num2);
    if(num1>num2){
        for(i=1;i<=num1;i++){
            if((num1%i==0) && (num2%i==0)){
                hcf = i;
            }
        }
    }
    else if(num1<num2){
          for(i=1;i<=num2;i++){
            if((num1%i==0) && (num2%i==0)){
                hcf = i;
            }
        }
    }
    else{
        hcf = num1;
    }
    lcm = (num1 * num2) / hcf ;
    printf("GCD = %d\nLCM = %d",hcf,lcm);
```

```
      return 0;
  }
```

Explanation :

int num1, num2, i, hcf, lcm;: Declare integer variables to store the two input numbers (num1 and num2), a loop counter (i), the highest common factor (hcf or GCD), and the least common multiple (lcm).

The program prompts the user to enter two positive integers using the printf() function and then reads the input using the scanf() function.

The program uses an if statement to check whether num1 is greater than num2, less than num2, or equal to num2.

Depending on whether num1 is greater or smaller than num2, the program uses a loop to find the highest common factor (GCD) of the two numbers. It iterates through numbers from 1 to the smaller of the two input numbers (num1 or num2). If both num1 and num2 are divisible by the loop counter i, then i is a common factor, and it is stored as the GCD.

The LCM is calculated using the formula (num1 * num2) / hcf, where hcf is the highest common factor (GCD) calculated earlier.

The program uses the printf() function to print the calculated GCD and LCM.

Problem:

6. Write a C program to display and find the sum of the series 1+11+111+....111 up to ***n.*** For e.g., if n=4, the series is: 1+11+111+1111. Take the value of 'n' as input from the user.

Code :

```
#include <stdio.h>
 int main() {
     int n;
     printf("Enter the value of n: ");
     scanf("%d", &n);
```

```
    int term = 1;
    int sum = 0;

    printf("Series: ");
    for (int i = 0; i < n; i++) {
        printf("%d", term);
        sum += term;
        if (i != n - 1) {
            printf(" + ");
        }
        term = term * 10 + 1;
    }

    printf("\nSum of the series: %d\n", sum);

    return 0;
}
```

Explanation :

int n;: Declares an integer variable n to store the user input for the number of terms in the series.

The program ask the user to enter the value of n (number of terms in the series) using the printf() function and then reads the input using the scanf() function.

int term = 1;: Initialize an integer variable term to 1. This variable represents the current term in the series.
int sum = 0;: Initialize an integer variable sum to 0. This variable will store the sum of the series.
Output: The program uses the printf() function to display the initial message "Series: " before printing the terms of the series.

The program uses a for loop to generate and print the terms of the series. The loop iterates from 0 to n - 1. In each iteration:

The current term is printed using printf().
The term is added to the sum.
If the current term is not the last term, the program prints " + " to separate terms.
The term is updated by multiplying it by 10 and adding 1. This generates the next term in the series.
Print sum: After the loop finishes, the program uses printf() to print the calculated sum of the series.

Problem:

7. Write a C program that reads a positive integer ***n*** and then prints the following pattern

```
*********
_********
__*******
___******
____*****
_____****
______***
_______**
________*
```

where ***n*** is the number of lines.

Code :

```
#include<stdio.h>
int main(){
    int i,j,n;
    printf("Enter the value of n : ");
    scanf("%d",&n);
    i=0;
    while(i<n){
            j=0;
            while(j<i){
            printf(" ");
            j++;
        }
        j=0;
        while(j<n-i){
            printf("*");
            j++;
        }


        printf("\n");


        i++;
    }
    return 0;
}
```

Explanation :

int i, j, n;: Declare integer variables i and j for loop counters and an integer variable n to store the user input for the number of rows in the inverted right triangle.
The program ask the user to enter the value of n (number of rows) using the printf() function and reads the input using the scanf() function.

i = 0;: Initialize the variable i to 0. This variable represents the current row index in the triangle.
The outer while loop runs as long as i is less than n. It controls the number of rows in the triangle.
Inner while loops: Inside the outer loop, there are two nested while loops. These loops control the printing of spaces (" ") and stars (*) within each row of the triangle.

j = 0; while (j < i): The first nested loop prints spaces before the stars. It runs i times, where i is the current row index. This creates the left indentation of the triangle.

j = 0; while (j < n - i): The second nested loop prints stars (*). It runs n - i times, where n - i is the number of stars in the current row.
Print newline: After each row is printed, the program uses printf("\n") to move to the next line and start a new row.

After completing a row, the variable i is incremented.

Problem:

8. Write a C program to find the reverse of an integer number.

Code :

```
#include <stdio.h>
int main() {
    int num, reversedNum = 0;

    printf("Enter an integer number: ");
    scanf("%d", &num);

    while (num != 0) {
        int digit = num % 10;
        reversedNum = reversedNum * 10 + digit;
        num /= 10;
    }

    printf("Reverse of the number: %d\n", reversedNum);

    return 0;
}
```

Explanation :

int num, reversedNum = 0;: Declare integer variables num to store the user input and reversedNum to store the reversed number.
Input: The program prompts the user to enter an integer number using the printf() function and reads the input using the scanf() function.

while (num != 0): This while loop runs as long as num is not equal to 0. It reverses the digits of the number.

int digit = num % 10;: Get the last digit of the number by taking the remainder of the number divided by 10.
reversedNum = reversedNum * 10 + digit;: Multiply the existing reversed number by 10 and then add the last digit. This effectively appends the last digit to the reversed number.
num /= 10;: Remove the last digit from the original number by integer division with 10.
After the loop finishes, the program uses printf() to print the reversed number.

Problem:

9. Write a C program to input ***n*** numbers in an array, calculate the sum of all even numbers and all odd numbers in the array and print the larger sum.

   Example:

   If the array contains the following elements:
   2, 3, 3, 5, 4, 8, 7, 11, 2
   The sum of all even elements is 2+4+8+2=16
   Sum of all odd elements is 3+3+5+7+11=29
   Therefore, the output should be 29.

Code :

```
#include<stdio.h>
int main(){
    int evenSum , oddSum,n,i,max;
    printf("Enter the value of n : ");
    scanf("%d",&n);
    int num[n];
    evenSum = 0;
    oddSum = 0;
    i=0;
    while(i<n){
        printf("\nEnter number %d : ",i+1);
        scanf("%d",&num[i]);
        if(num[i]%2==0){
            evenSum +=num[i];
        }
        else{
            oddSum +=num[i];
        }



        i++;
    }

    if(evenSum>oddSum){
    max = evenSum;
    }
    else{
    max = oddSum;
    }
    printf("\nodd sum is : %d ",oddSum);
    printf("\neven sum is : %d ",evenSum);
    printf("\nlarger sum is : %d ",max);



    return 0;
}
```

Explanation :

int evenSum, oddSum, n, i, max;: Declare integer variables to store the sum of even numbers (evenSum), the sum of odd numbers (oddSum), the number of elements (n), a loop counter (i), and the maximum of the two sums (max).

int num[n];: Declare an integer array num with a size of n to store the user input numbers.

The program prompts the user to enter the value of n (number of elements) using the printf() function and reads the input using the scanf() function.

evenSum = 0;: Initialize evenSum to 0 to keep track of the sum of even numbers.

oddSum = 0;: Initialize oddSum to 0 to keep track of the sum of odd numbers.

i = 0;: Initialize the loop counter i to 0.

The program uses a while loop to get input from the user for n numbers. In each iteration of the loop:

The program prompts the user to enter a number using printf() and reads the input using scanf().

It checks whether the entered number is even or odd using the modulo operator (num[i] % 2) and updates the corresponding sum (evenSum or oddSum).

Determine the larger sum: After collecting input and calculating sums, the program compares evenSum and oddSum to determine which sum is larger and assigns the larger value to max.

The program uses printf() to print the sums of even and odd numbers.

It also prints the larger of the two sums.

Problem:

10. Write a C program to print the following pattern:

```
a) 1           b) 1
   1 2            2 2
   1 2 3          3 3 3
   1 2 3 4        4 4 4 4
   1 2 3 4 5      5 5 5 5 5
```

Code (a):

```
#include<stdio.h>
int main(){
    int i,j,n;
    printf("Enter the value of n : ");
    scanf("%d",&n);
    i=1;

    while(i<=n){
        j=1;
        while(j<=i){
            printf("%d ",i);
            j++;
        }
        printf("\n");
        i++;
    }
    return 0;
}
```
Explanation :

int i, j, n;: Declare integer variables i and j for loop counters and an integer variable n to store the user input for the number of rows in the triangle.
The program prompts the user to enter the value of n (number of rows) using the printf() function and reads the input using the scanf() function.

i = 1;: Initialize the variable i to 1. This variable represents the current row index in the triangle.
Outer while loop: The outer while loop runs as long as i is less than or equal to n. It controls the number of rows in the triangle.

Inside the outer loop, there is another nested while loop. This loop controls the printing of numbers within each row of the triangle.

j = 1; while (j <= i): This nested loop prints the value of i j times in the same row.
Print newline: After each row is printed, the program uses printf("\n") to move to the next line and start a new row.

After completing a row, the variable i is incremented.

Code(b) :

```
#include<stdio.h>
int main(){
    int i,j,n;
    printf("Enter the value of n : ");
    scanf("%d",&n);
    i=1;

    while(i<=n){

        j=1;
        while(j<=i){
            printf("%d ",j);
            j++;
        }


        printf("\n");


        i++;
    }
    return 0;
}
```

Explanation :

int i, j, n;: Declare integer variables i and j for loop counters and an integer variable n to store the user input for the number of rows in the triangle.
Input: The program prompts the user to enter the value of n (number of rows) using the printf() function and reads the input using the scanf() function.

i = 1;: Initialize the variable i to 1. This variable represents the current row index in the triangle.
Outer while loop: The outer while loop runs as long as i is less than or equal to n. It controls the number of rows in the triangle.

Inside the outer loop, there is another nested while loop. This loop controls the printing of numbers within each row of the triangle.

j = 1; while (j <= i): This nested loop prints the value of j j times in the same row. Since j increases with each iteration of the inner loop, the numbers printed in each row are sequentially increasing from 1.

After each row is printed, the program uses printf("\n") to move to the next line and start a new row.

After completing a row, the variable i is incremented.