**Problem Solving**

**Course**: Structured Programming in C

**Author**: Abeng Christian Nguh

Below, will be solution to various problems and explanation of codes. The codes are written and compile in ubuntu.

Section: A

**5**. Write a C program to check whether a 3-digit number is a magic number or

not. (Palindrome) A number is a magic number if its reverse is same as the

original number. (filename: magic\_number.c)

#include <stdio.h>

int main() {

int number, originalNumber, reversedNumber = 0, remainder;

printf("Enter a 3-digit number: ");

scanf("%d", &number);

originalNumber = number;

// Reverse the number

while (number != 0) {

remainder = number % 10;

reversedNumber = reversedNumber \* 10 + remainder;

number /= 10;

}

if (originalNumber == reversedNumber) {

printf("%d is a magic number (palindrome).\n", originalNumber);

} else {

printf("%d is not a magic number.\n", originalNumber);

}

return 0;

}

**Explanation:**

* We take input from the user for a 3-digit number.
* We store the original number for later comparison.
* We extract the last digit using the modulus operator.
* We remove the last digit by dividing the number by 10.
* We calculate the sum of the remaining digits.
* Finally, we compare the sum with the original number to determine if it’s a magic number.

**6.** Any character is entered through the keyboard, Write a C program to

determine whether the character entered is a capital letter, a small case letter,

a digit or a special symbol. (filename: character.c)

#include <stdio.h>

int main() {

char ch;

// Read a character from the user

printf("Enter a character: ");

scanf("%c", &ch);

// Check if the character is an uppercase letter

if (ch >= 'A' && ch <= 'Z') {

printf("%c is an uppercase letter.\n", ch);

}

// Check if the character is a lowercase letter

else if (ch >= 'a' && ch <= 'z') {

printf("%c is a lowercase letter.\n", ch);

}

// Check if the character is a digit

else if (ch >= '0' && ch <= '9') {

printf("%c is a digit.\n", ch);

}

// If none of the above conditions are true, it is a special symbol

else {

printf("%c is a special symbol.\n", ch);

}

return 0;

}

**Explanation:**

* We declare a variable ch to store the input character.
* The printf statement prompts the user to enter a character.
* The scanf function reads a character from the user.
* We use conditional statements (if, else if, and else) to determine the type of character based on its ASCII value:
  + If the ASCII value falls within the range of uppercase letters ('A' to 'Z') or lowercase letters ('a' to 'z'), we classify it as an alphabet.
  + If the ASCII value is between '0' and '9', it’s a digit.
  + Otherwise, it’s considered a special character.

1. Write a C program to print the table of n. (filename: time\_table.c)

#include <stdio.h>

int main() {

int n, i;

// Input the value of 'n'

printf("Enter a number: ");

scanf("%d", &n);

// Print the table of 'n'

printf("Table of %d:\n", n);

for (i = 1; i <= 10; i++) {

printf("%d x %d = %d\n", n, i, n \* i);

}

return 0;

}

**Explanation:**

* We declare an integer variable **n** to store the user input.
* The printf statement prompts the user to enter an integer.
* The scanf function reads the user input and stores it in **n**.
* The for loop runs from **i = 1** to **i = 10**.
* Inside the loop, we calculate the product of **n** and **i** using **n \* i** and print the result.

8. Write a function that returns the value of x raised to the power of y. (filename:

power.c)

* - Prototype: int \_pow\_recursion(int x, int y);
* - If y is lower than 0, the function should return -1

#include <stdio.h>

// Recursive function to calculate x raised to the power of y

int \_pow\_recursion(int x, int y) {

// Base case: if y is 0, return 1 (x^0 = 1)

if (y == 0)

return 1;

// If y is negative, return -1 (since we don't handle negative exponents)

if (y < 0)

return -1;

// Recursive case: multiply x by the result of (\_pow\_recursion(x, y-1))

return x \* \_pow\_recursion(x, y - 1);

}

int main() {

int base, exponent;

printf("Enter the base (x): ");

scanf("%d", &base);

printf("Enter the exponent (y): ");

scanf("%d", &exponent);

int result = \_pow\_recursion(base, exponent);

if (result == -1)

printf("Error: Exponent cannot be negative.\n");

else

printf("%d^%d = %d\n", base, exponent, result);

return 0;

}

**Explanation:**

* The \_pow\_recursion function is defined to calculate **x** raised to the power of **y**.
* The base case checks if **y** is 0, in which case it returns 1 (since any number raised to the power of 0 is 1).
* If **y** is negative, the function returns -1 (as specified).
* Otherwise, it recursively multiplies **x** by the result of \_pow\_recursion(x, y - 1).