Function Pointer assignments

1. WAP to invoke functions below based on user input character using function pointer. Character mapping and respective functions to be invoked are given below.

Character input Function
+ int add(int x, int y)
- int sub(int x, int y)
* int multiply(int x, int y)
/ int divide(int x, int y)

Sol:

```
#include <stdio.h>

int add(int x, int y);

int sub(int x, int y);

int multiply(int x, int y);

int divide(int x, int y);


int main() {
   int (*operation)(int, int);

   char operator;

   int num1, num2;


   printf("Enter operator (+, -, *, /): ");

   scanf("%c", &operator);


  printf("Enter two integers: ");

   scanf("%d %d", &num1, &num2);


   switch (operator) {

      case '+':

         operation = add;

         break;
```

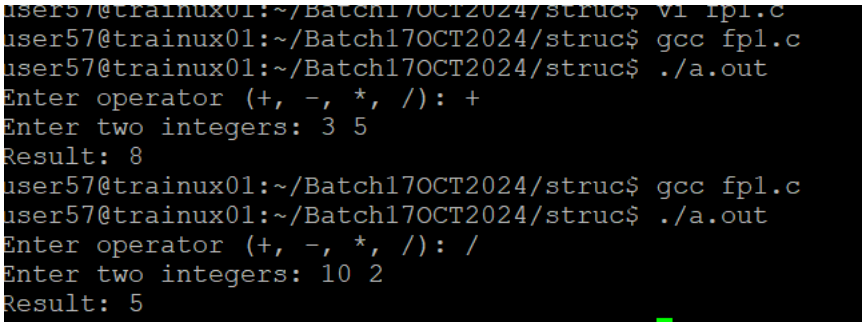
```
        case '-':
            operation = sub;
            break;
        case '*':
            operation = multiply;
            break;
        case '/':
            operation = divide;
            break;
        default:
            printf("Invalid operator\n");
            return 1;
    }

    int result = operation(num1, num2);
    printf("Result: %d\n", result);
return 0;
}

int add(int x, int y) {
    return x + y;
}
int sub(int x, int y) {
    return x - y;
}
int multiply(int x, int y) {
    return x * y;
}
int divide(int x, int y) {
    if (y == 0) {
        printf("Error! Division by zero.\n");
```

```
    return 0;
  }
  return x / y;
}
```

Output:



2. Extend Q1 to include a function below which will return the address of a function based on input character.
<address of function> getaddr(char mychar);

Sol:

```
#include <stdio.h>

int add(int x, int y);

int sub(int x, int y);

int multiply(int x, int y);

int divide(int x, int y);


int (*getaddr(char mychar))(int, int);


int main() {


  int (*operation)(int, int);

  char operator;

  int num1, num2;
```

```
    printf("Enter operator (+, -, *, /): ");
    scanf(" %c", &operator);

    printf("Enter two integers: ");
    scanf("%d %d", &num1, &num2);
   operation = getaddr(operator);

   if (operation == NULL) {
       printf("Invalid operator\n");
       return 1;
    }
    int result = operation(num1, num2);
    printf("Result: %d\n", result);
  return 0;
}
int add(int x, int y) {
    return x + y;
}
int sub(int x, int y) {
    return x - y;
}
int multiply(int x, int y) {
    return x * y;
}
int divide(int x, int y) {
    if (y == 0) {
       printf("Error! Division by zero.\n");
       return 0;
    }
    return x / y;
}
```

```
int (*getaddr(char mychar))(int, int) {
  switch (mychar) {
    case '+':
      return add;
    case '-':
      return sub;
    case '*':
      return multiply;
    case '/':
      return divide;
    default:
      return NULL;
  }
}
```

Output:

```
user57@trainux01:~/Batch17OCT2024/struc$ vi fp2.c
user57@trainux01:~/Batch17OCT2024/struc$ gcc fp2.c
user57@trainux01:~/Batch17OCT2024/struc$ ./a.out
Enter operator (+, -, *, /): -
Enter two integers: 10 6
Result: 4
```

3. Extend Q1 to include a function below which takes a function pointer to a calculator function and one integer (value = 10) as arguments and shall invoke the given function with required arguments. For the second argument read input from user. Return the result.
int invokefunc(<function pointer as argument1>, int val1);

Sol:

```
#include <stdio.h>

int add(int x, int y);

int sub(int x, int y);

int multiply(int x, int y);

int divide(int x, int y);
```

```
int (*getaddr(char mychar))(int, int);
int invokefunc(int (*func_ptr)(int, int), int val1);
int main() {
    int (*operation)(int, int);
   char operator;
   int result;
   printf("Enter operator (+, -, *, /): ");
   scanf(" %c", &operator);

   operation = getaddr(operator);
   if (operation == NULL) {
      printf("Invalid operator\n");
      return 1;
   }
  result = invokefunc(operation, 10);
   printf("Result: %d\n", result);

   return 0;
}
int add(int x, int y) {
   return x + y;
}
int sub(int x, int y) {
   return x - y;
}
int multiply(int x, int y) {
   return x * y;
}
int divide(int x, int y) {
   if (y == 0) {
      printf("Error! Division by zero.\n");
```

```
        return 0;
    }
    return x / y;
}

int (*getaddr(char mychar))(int, int) {
    switch (mychar) {
        case '+':
            return add;
        case '-':
            return sub;
        case '*':
            return multiply;
        case '/':
            return divide;
        default:
            return NULL;
    }
}

int invokefunc(int (*func_ptr)(int, int), int val1) {
    int val2;
    printf("Enter the second integer: ");
    scanf("%d", &val2);
     return func_ptr(val1, val2);
}
```

Output:

```
user57@trainux01:~/Batch17OCT2024/struc$ vi fp3.c
user57@trainux01:~/Batch17OCT2024/struc$ gcc fp3.c
user57@trainux01:~/Batch17OCT2024/struc$ ./a.out
Enter operator (+, -, *, /): *
Enter the second integer: 4
Result: 40
```

4. Extend Q1 to define an array of function pointers and invoke them based user input character.

Sol:

```
#include <stdio.h>

int add(int x, int y);

int sub(int x, int y);

int multiply(int x, int y);

int divide(int x, int y);


int invokefunc(int (*func_ptr)(int, int), int val1);


int main() {
   int (*operation[])(int, int) = {add, sub, multiply, divide};
   char operator;
   int result;


   printf("Enter operator (+, -, *, /): ");
   scanf(" %c", &operator);


   int val1 = 10;
   int index = -1;


   switch (operator) {
      case '+': index = 0; break;
      case '-': index = 1; break;
      case '*': index = 2; break;
      case '/': index = 3; break;
      default:
         printf("Invalid operator\n");
         return 1;
```

```
    }

    result = invokefunc(operation[index], val1);
    printf("Result: %d\n", result);

    return 0;
}

int add(int x, int y) {
    return x + y;
}

int sub(int x, int y) {
    return x - y;
}

int multiply(int x, int y) {
    return x * y;
}

int divide(int x, int y) {
    if (y == 0) {
        printf("Error! Division by zero.\n");
        return 0;
    }
    return x / y;
}

int invokefunc(int (*func_ptr)(int, int), int val1) {
    int val2;
    printf("Enter the second integer: ");
```

```
    scanf("%d", &val2);
    return func_ptr(val1, val2);
}
```

Output: