Assignment 1

Neil Kamal Dhami - EE18BTECH11031

Download all latex-tikz codes from

```
https://github.com/neildhami18/
   IITH_Academics/blob/main/EE4013/
   Assignment1/Assignment1.tex
```

1 Problem

(Q 18) Consider the following C program.

```
#include <stdio.h>
int jumble(int x, int y){
   y = 2*x + y;
   return y;
}
int main(){
   int x=2, y=5;
   y = jumble(y,x);
   x = jumble(y,x);
   printf("%d \n", x);
   return 0;
}
```

The value printed by the program is?

2 Solution

Answer : 26

Explanation:
This is a very simple function call problem. The only tricky part involved is that the global variables **x** and **y** are called as parameters y and x respectively during function call. The function calls in the main loop return the values for **x** and **y** as:

```
   y = jumble(y,x) = 2*5 + 2 = 12
   x = jumble(y,x) = 2*12 + 2 = 26
```

3 Concept

The function **jumble** performs an operation commonly known as "**saxpy**" (**S**ingle-precision real **A**lpha **X** **P**lus **Y**)

Definition 1 (Saxpy)**.** *If x, $y \in \mathbb{R}^n$ and $a \in \mathbb{R}$, then this operation overwrites y with $y + a*x$.*

This function can also be presented as a dot product of a coefficient vector with the input vector:

$$y = \begin{pmatrix} a & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \tag{3.0.1}$$

where a = 2 in our case.

4 Application

Let us utilise data structures to save two vectors and develop a function to generate their dot product.
We employ linked lists to store our data since they are dynamic Data Structures and allow utilisation of simple insertion/deletion functions.
Linked List Data Structure:

```
   struct node {
      int data;
      int key;
      struct node *next;
   };
```

Given repository contains full code.

```
https://github.com/neildhami18/
   IITH_Academics/blob/main/EE4013/
   Assignment1/codes/dot_product.c
```

Glimpse of the function performing the dot operation of two linked lists:

```
int dot(struct node* List1, struct node*
    List2)
{
   int product = 0;
   struct node *current_a = List1;
   struct node *current_b = List2;

   while(current_a != 0 && current_b
      !=0)
   {
      if(current_a->key == current_b->
         key)
```

```
        {
            product = product + current_a
                ->data * current_b->data;
            current_a=current_a->next;
            current_b=current_b->next;
        }
        else if(current_a->key <
            current_b->key)
        {
            current_a=current_a->next;
        }
        else
        {
            current_b=current_b->next;
        }
    }
    return product;
}
```

5 File Handling

I have also added a code segment(extention) in order to facilitate reading of vector data from text files into the linked lists.
File Handling Code Snippet:

```
    f = fopen("vector.txt", "r");
      int i=1;
      while(fgets(line, sizeof(line), f)
          )
      {
          LIST *node = malloc(sizeof(
              LIST));
          node->string = strdup(line);
          int data = atoi(node->string);
          List = insertFirst(List,i,data
              );
          i++;
      }
      fclose(f);
```