FEQ News

FEQ1

Any questions from previous OLQs

05/04 6PM – 05/05 11:59PM

FEQ2

File Handling with structs, malloc and pointers

05/05 6PM – 05/06 11:59PM

FEQ3

Linked Lists

05/07 6PM – 05/08 11:59PM

FEQ4

Stacks and Queues

05/08 6PM – 05/09 11:59PM

FEQ5

Binary Search Trees

05/09 6PM – 05/10 11:59PM

CSE 1320

Week of 05/01/2023

Instructor : Donna French

FEQ1

- Questions from previous OLQs
- Base conversions
- Recursion
- Command Line Arguments
- Debug
- String Handling
- Two's Complement
- Bitwise Operators

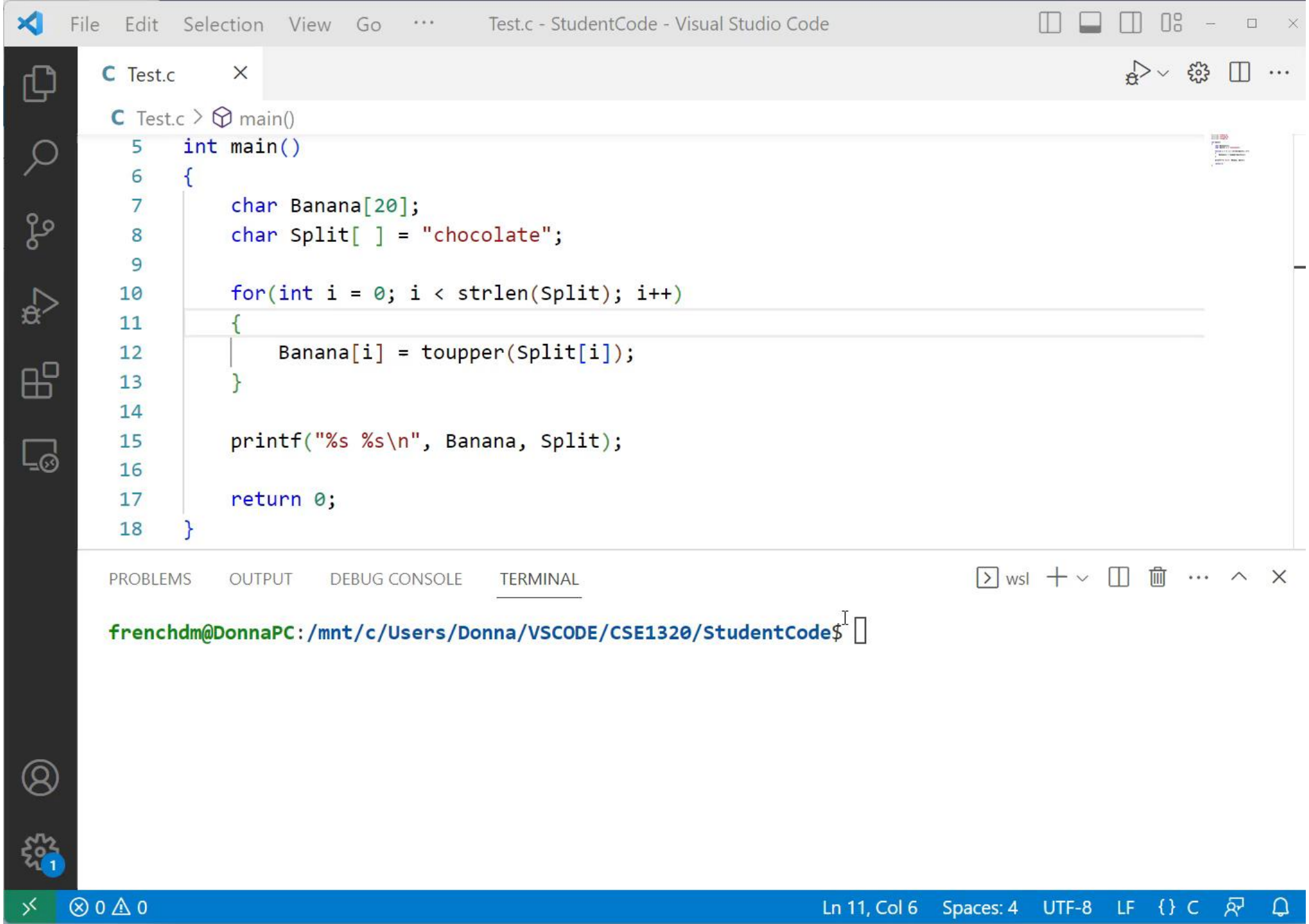
Write the following code

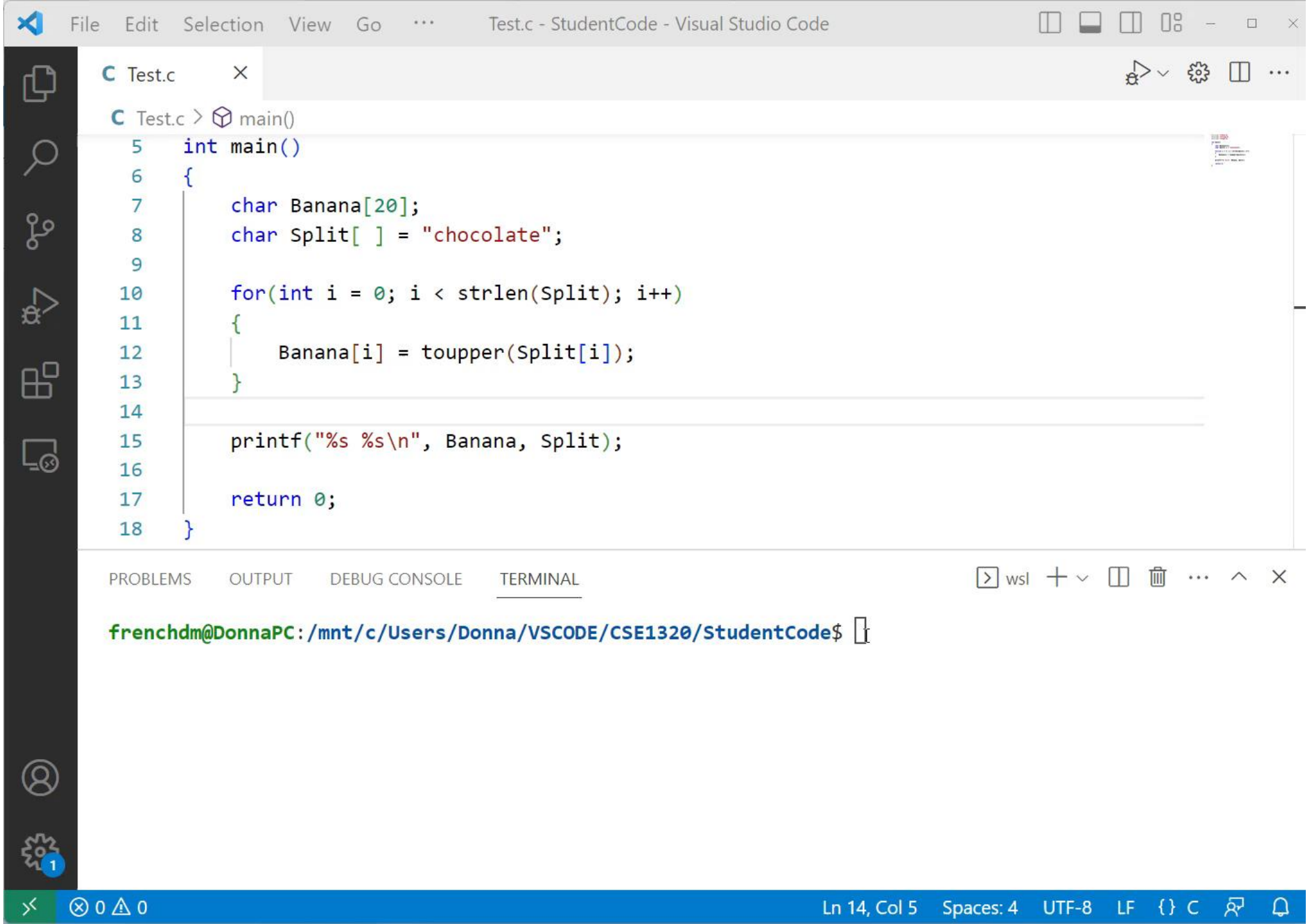
1. Declare a char array named Banana
2. Declare a char array named Split and set it to "chocolate"
3. Store the uppercase version of Split in Banana
4. Print the strings Banana and Split

Write the following code

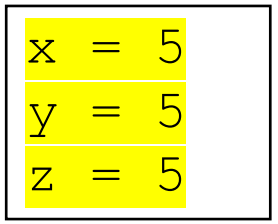
1. Declare a char array named Banana
2. Declare a char array named Split and set it to "chocolate"
3. Store the uppercase version of Split in Banana
4. Print the strings Banana and Split

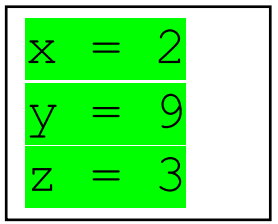
```c
#include <stdio.h>
#include <string.h>
#include <ctype.h>

int main()
{
    char Banana[20];
    char Split[ ] = "chocolate";

    for(int i = 0; i < strlen(Split); i++)
    {
        Banana[i] = toupper(Split[i]);
    }

    printf("%s %s\n", Banana, Split);

    return 0;
}
```

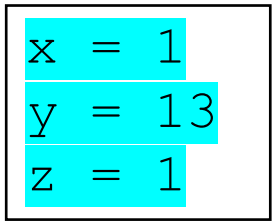

File Edit Selection View Go ···
Test.c - StudentCode - Visual Studio Code
Test.c
Test.c > main()
5 int main()
6 {
7 char Banana[20];
8 char Split[ ] = "chocolate";
9
10 for(int i = 0; i < strlen(Split); i++)
11 {
12 Banana[i] = toupper(Split[i]);
13 }
14
15 printf("%s %s\n", Banana, Split);
16
17 return 0;
18 }
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
wsl
frenchdm@DonnaPC:/mnt/c/Users/Donna/VSCODE/CSE1320/StudentCode$
0 0
Ln 11, Col 6 Spaces: 4 UTF-8 LF { } C

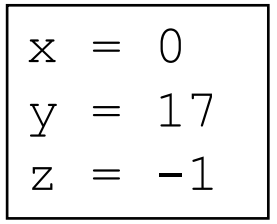

File Edit Selection View Go ···
Test.c - StudentCode - Visual Studio Code
Test.c
Test.c > main()
5 int main()
6 {
7 char Banana[20];
8 char Split[ ] = "chocolate";
9
10 for(int i = 0; i < strlen(Split); i++)
11 {
12 Banana[i] = toupper(Split[i]);
13 }
14
15 printf("%s %s\n", Banana, Split);
16
17 return 0;
18 }
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
wsl
frenchdm@DonnaPC:/mnt/c/Users/Donna/VSCODE/CSE1320/StudentCode$
0 0
Ln 14, Col 5 Spaces: 4 UTF-8 LF { } C

```c
#include <stdio.h>

void Hello(int x, int y, int z)
{
    if (z < 1)
        return;

    printf("%d%d%d\n", x, y, z);

    Hello(x/2, y+4, z-2);

    printf("%d%d%d\n", z, y, x);
}

int main()
{

    int x = 5, y = 5, z = 5;

    Hello(x, y, z);
}
```









555

293

1131

1131

392

555

```c
#include <stdio.h>
#include <string.h>

int main(int argc, char *argv[])
{
   int count = 0;

   if (argc > 1)
   {
      for (int i = 1; i < argc; i++)
      {
         if (!strcmp(argv[i], "OLQ"))
         {
            count++;
         }
      }
   }

   printf("%d", count);

   return 0;
}
```

```
./a.out ROLQ OLQ ABC OLQ13
1

./a.out ROLQ olq ABC OLQ13
0

./a.out OLQ ABC OLQ DEF
2

./a.out olq ABC OLQ DEF
1
```

FEQ2

File handling with

structs

`malloc()`

pointers

Using dynamic memory allocation to read a file with variable length fields.

```
1 Tacos|Flamin' Hot Doritos Locos Taco|Seasoned Beef, Cheese, Lettuce
2 Tacos|Flamin' Hot Doritos Locos Tacos Supreme|Seasoned Beef, Cheese, Lettuce, Tomatoes, Sour Cream
3 Burritos|Loaded Taco Grande Burrito|Seasoned Beef, Cheese, Lettuce, Red Strips, Sour Cream
```

```
typedef struct
{
   char *category;
   char *name;
   char *whatsincluded;
}
TACOBELL;

int main(int argc, char *argv[])
{
   TACOBELL Menu[20] = {};
   char *token = NULL;
   char filename[20] = {};
   FILE *FH = NULL;
   char FileLine[200];
   int MenuCount = 0;
   int i;
```

```
strcpy(filename, argv[1]);
FH = fopen(filename, "r+");

if (FH == NULL)
{
   printf("File did not open");
   exit(0);
}
```

```
1 Tacos|Flamin' Hot Doritos Locos Taco|Seasoned Beef, Cheese, Lettuce
2 Tacos|Flamin' Hot Doritos Locos Tacos Supreme|Seasoned Beef, Cheese, Lettuce, Tomatoes, Sour Cream
3 Burritos|Loaded Taco Grande Burrito|Seasoned Beef, Cheese, Lettuce, Red Strips, Sour Cream
```

```
while (fgets(FileLine, sizeof(FileLine)-1, FH))
{
   token = strtok(FileLine, "|");
   Menu[MenuCount].category = malloc(strlen(token)*sizeof(char)+1);
   strcpy(Menu[MenuCount].category, token);

   token = strtok(NULL, "|");
   Menu[MenuCount].name = malloc(strlen(token)*sizeof(char)+1);
   strcpy(Menu[MenuCount].name, token);

   token = strtok(NULL, "|");
   Menu[MenuCount].whatsincluded = malloc(strlen(token)*sizeof(char)+1);
   strcpy(Menu[MenuCount].whatsincluded, token);

   MenuCount++;
}
```

```
1 Tacos|Flamin' Hot Doritos Locos Taco|Seasoned Beef, Cheese, Lettuce
2 Tacos|Flamin' Hot Doritos Locos Tacos Supreme|Seasoned Beef, Cheese, Lettuce, Tomatoes, Sour Cream
3 Burritos|Loaded Taco Grande Burrito|Seasoned Beef, Cheese, Lettuce, Red Strips, Sour Cream
```

```
for (i = 0; i < MenuCount; i++)
{
   printf("Category : %s\nName      : %s\n\nWhat's Included : %s\n\n",
      Menu[i].category, Menu[i].name, Menu[i].whatsincluded);
}
```

```
for (i = 0; i < MenuCount; i++)
{
   free(Menu[i].category);
   free(Menu[i].name);
   free(Menu[i].whatsincluded);
}
```

```
1 Tacos|Flamin' Hot Doritos Locos Taco|Seasoned Beef, Cheese, Lettuce
2 Tacos|Flamin' Hot Doritos Locos Tacos Supreme|Seasoned Beef, Cheese, Lettuce, Tomatoes, Sour Cream
3 Burritos|Loaded Taco Grande Burrito|Seasoned Beef, Cheese, Lettuce, Red Strips, Sour Cream
```

FEQ2

For each line of code labeled with a letter, match the number of the line of code that should be in that location.

Use the comments to pick the correct ordering - the order of the lines must not only work as a program, but they also must match the comments.

For example, line A matches up with line 1 from the list of code lines. You would put 1 in blank A for the answer.

FEQ3

Linked Lists

Given code that forms a linked list, can you do the following

- add a node to the start of the linked list
- display the linked list
- add a node to the end of the linked list
- delete a node from the linked list

```
void AddNode(int NewNodeNumber, node **LinkedListHead)
{
   node *TempPtr, *NewNode;

   NewNode = malloc(sizeof(node));
   NewNode->node_number = NewNodeNumber;
   NewNode->next_ptr = NULL;

   if (*LinkedListHead == NULL)
   {
      *LinkedListHead = NewNode;
   }
   else
   {
      TempPtr = *LinkedListHead;

      while (TempPtr->next_ptr != NULL)
         TempPtr = TempPtr->next_ptr;

      TempPtr->next_ptr = NewNode;
   }
}
```

```
void DisplayLinkedList(node *LinkedListHead)
{
   node *TempPtr = LinkedListHead;

   while (TempPtr != NULL)
   {
      printf("\nNode Number %d\t\tNode Address %p     Node Next Pointer %p\n",
      TempPtr->node_number, TempPtr, TempPtr->next_ptr);
      TempPtr = TempPtr->next_ptr;
   }
}
```

FEQ3

FEQ4

Stack

Queue

FEQ4

Stack

Given the code that forms a stack, can you do the following

- push a node
- pop a node
- display the stack

FEQ4

```
void pop(SNODE **StackTop)
{
    if (*StackTop != NULL)
    {
        SNODE *TempPtr = (*StackTop)->next_ptr;

        free(*StackTop);
        *StackTop = TempPtr;
    }
}
```

```
void push(SNODE **StackTop, LNODE *TicketList, int ReceiptNumber, char MTN[])
{
   SNODE *NewNode = malloc(sizeof(SNODE));
   NewNode->ReceiptNumber = ReceiptNumber;
   NewNode->TicketList =  TicketList;
   NewNode->MovieTheaterName = malloc(strlen(MTN) * sizeof(char) + 1);
   strcpy(NewNode->MovieTheaterName, MTN);
   NewNode->next_ptr = NULL;

   if (*StackTop == NULL)
   {
      *StackTop = NewNode;
   }
   else
   {
      NewNode->next_ptr = *StackTop;
      *StackTop = NewNode;
   }
}
```

FEQ4

FEQ4

Queue

Given the code that forms a queue, can you do the following

- enqueue
- dequeue
- display the queue

```
void enQueue(char CustomerName[], QNODE **QueueHead, QNODE **QueueTail)
{
   QNODE *NewNode = malloc(sizeof(QNODE));
   NewNode->name = malloc(strlen(CustomerName) * sizeof(char) + 1);
   strcpy(NewNode->name, CustomerName);
   NewNode->next_ptr = NULL;

   if (*QueueHead == NULL)
   {
      *QueueHead = *QueueTail = NewNode;
   }
   else
   {
      (*QueueTail)->next_ptr = NewNode;
      *QueueTail = NewNode;
   }
}
```

FEQ4

FEQ4

```
void deQueue(QNODE **QueueHead)
{
   QNODE *TempPtr = (*QueueHead)->next_ptr;

   if (*QueueHead != NULL)
   {
      free(*QueueHead);
      *QueueHead = TempPtr;
   }
}
```

```
void DisplayQueue(QNODE *QueueHead)
{
   QNODE *TempPtr = QueueHead;

   while (TempPtr != NULL)
   {
      printf("%s\n", TempPtr->name);
      TempPtr =  TempPtr->next_ptr;
   }
}
```

FEQ5
Binary Search Tree

Given the lines of code needed to form the function to add a node to the binary search tree, can you put the code lines in the correct order to correctly form the function?

Given the lines of code needed to form the function to traverse a binary search tree, can you put the code lines in the correct order to correctly form the function?

```
void AddBSTNode(BNODE **BSTnode, char MTN[], char ZC[], char FN[], char DIM[])
{
   if (*BSTnode == NULL)
   {
      *BSTnode = malloc(sizeof(BNODE));
      (*BSTnode)->left = (*BSTnode)->right = NULL;
      (*BSTnode)->MovieTheaterName = malloc(strlen(MTN) * sizeof(char) + 1);
      strcpy((*BSTnode)->MovieTheaterName, MTN);
      strcpy((*BSTnode)->ZipCode, ZC);
      (*BSTnode)->FileName = malloc(strlen(FN) * sizeof(char) + 1);
      strcpy((*BSTnode)->FileName, FN);
      strcpy((*BSTnode)->Dimensions, DIM);
   }
   else
   {
      if (strcmp(ZC, (*BSTnode)->ZipCode) < 0)
         AddBSTNode(&(*BSTnode)->left, MTN, ZC, FN, DIM);
      else if(strcmp(ZC, (*BSTnode)->ZipCode) > 0)
         AddBSTNode(&(*BSTnode)->right, MTN, ZC, FN, DIM);
      else
         printf(" Duplicate Element !! Not Allowed !!!");
   }
}
```

FEQ5

```
void InOrder(BNODE *bnode)
{
   if(bnode != NULL)
   {
      InOrder(bnode->left);
```

```
void PostOrder(BNODE *bnode)
{
   if(bnode != NULL)
   {
      PostOrder(bnode->left);
      PostOrder(bnode->right);
      printf("%-40s %5s\n", bnode->MovieTheaterName, bnode->ZipCode);
   }
}
```

```
      printf("%-40s %5s\n", bnode->MovieTheaterName, bnode->ZipCode);
      PreOrder(bnode->left);
      PreOrder(bnode->right);
   }
}
```

FEQ5