

Exams

Programming Concepts

Peter M. Maurer



NAME _____

1. The following program has (at least) 10 syntax errors. Circle each error. Write the corrected program in the blank space below. 2 points for each error you find. -2 points for circling something that's OK. (20 points total.)

```
#include stdio.h
```

```
void main(  
{  
    int j;k;  
  
    j = 0;;  
    for (j>10 ; do j++)  
    {  
        if j==7 then  
        {  
            printf(j is equal to 7);  
        }  
        else  
        {  
            k = j  
        }  
    }  
    exit 100;  
}
```

2. Using the integer variables x, y, and z, write a single statement for each of the following tasks. (2 points each, 20 points total.)

- a) Square x, place result in y

- b) Divide y by 17, place result in z.

- c) Add 1 to x (leave result in x)

- d) Divide z by 7, leave remainder in x. (Discard quotient)

- e) Multiply y by 3. (leave result in y)

- f) Add x, y, and z together, leave result in x

- g) Store a 1 in x if y is greater than z, store a 0 in x otherwise

- h) Subtract x from y, leave result in y.

- i) Change the sign of z. (Leave result in z.)

- j) Subtract x from y, leave result in z.

4. Write single if statements (if/else OK) to do the following tasks. (4 points each, 20 points total.)

a) Add x to z if y is greater than ten.

b) Print "X is odd" or "X is even" depending on whether x is odd or even.
(Remainder by 2).

c) Print "String is too long" if the string MyString has more than 20 characters, not counting the '\0' character.

d) Print "File is at end" if the end-of-file has been encountered for the file MyFile.

e) Add 1 to x if y is equal to zero, add 2 to x otherwise.

NAME _____

1. The following program has (at least) 10 syntax errors. Circle each error. Write the corrected program in the blank space below. 2 points for each error you find. -2 points for circling something that's OK. (20 points total.)

```
#include stdio.h

void main(
{
    int j;k;

    j = 0;;
    for (j>10 ; do j++)
    {
        if j==7 then
        {
            printf(j is equal to 7);
        }
        else
        {
            k = j
        }
    }
    exit 100;
}
```

```
#include <stdio.h> // missing angle-brackets

void main( ) // missing right paren
{
    int j,k; // comma between j and k

    j = 0;;
    for ( ; j>10 ; j++) // two semicolons needed
                        // do must be removed
    {
        if (j==7) // incorrect if syntax
        {
            printf("j is equal to 7"); // quotes needed
        }
        else
        {
            k = j; // semicolon required
        }
    } // closing brace missing
    exit(100); // missing parens around argument
}
```

2. Using the integer variables x, y, and z, write a single statement for each of the following tasks. (2 points each, 20 points total.)

a) Square x, place result in y

y = x * x;

b) Divide y by 17, place result in z.

z = y / 17;

c) Add 1 to x (leave result in x)

x++;

d) Divide z by 7, leave remainder in x. (Discard quotient)

x = z % 7;

e) Multiply y by 3. (leave result in y)

y *= 3;

f) Add x, y, and z together, leave result in x

x += y + z;

g) Store a 1 in x if y is greater than z, store a 0 in x otherwise

x = y > z;

h) Subtract x from y, leave result in y.

y -= x;

i) Change the sign of z. (Leave result in z.)

z = -z;

j) Subtract x from y, leave result in z.

z = y - x;

3. Write for statements to do the following. Use the integer variables x, y, and z in addition to the loop index i (if necessary). (4 points each, 20 points total).

- a) Add up the integers from 3 to 17 inclusive.

```
for (i=3,x=0 ; i<18 ; i++)  
{  
    x += i;  
}
```

- b) Add up the EVEN integers from 3 to 17 inclusive.

```
for (i=4,x=0 ; i<17 ; i+=2)  
{  
    x += i;  
}
```

- c) Add up the digits of x, base 10. The result for 123 would be 6.

```
for (y=0 ; x>0 ; x /= 10)  
{  
    y += x % 10;  
}
```

- d) Print numbers that are powers of 2, starting with 1 and ending with 2048. DO NOT use a loop index.

```
for (x=1 ; x<2049 ; x *= 2)  
{  
    printf("%d\n",x);  
}
```

- e) Print the characters of the string MyString, one character per line.

```
for (x=0 ; MyString[x] ; x++)  
{  
    printf("%c\n",MyString[x]);  
}
```

4. Write single if statements (if/else OK) to do the following tasks. (4 points each, 20 points total.)

a) Add x to z if y is greater than ten.

```
if (y>10)  
{  
    z += x;  
}
```

b) Print “X is odd” or “X is even” depending on whether x is odd or even.
(Remainder by 2).

```
if (x%2)  
{  
    printf("X is odd\n");  
}  
else  
{  
    printf("X is even\n");  
}
```

c) Print “String is too long” if the string MyString has more than 20 characters, not counting the ‘\0’ character.

```
if (strlen(MyString) > 20)  
{  
    printf("String is too long\n");  
}
```

d) Print “File is at end” if the end-of-file has been encountered for the file MyFile.

```
if (feof(MyFile))  
{  
    printf("File is at end\n");  
}
```

e) Add 1 to x if y is equal to zero, add 2 to x otherwise.

```
if (y == 0)  
{  
    x++;  
}  
else  
{  
    x += 2;  
}
```

5. Write a whole program that reads one line of text containing last name, first name, and middle initial (without period), in that order. Assume that the three parts of the name are separated by spaces. Print the name in the following order and quit: First Name, Middle Initial (with period), Last Name. (10 points)

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

void main()
{
    char First[20],Middle[20],Last[20];

    scanf("%s %s %s",Last,First,Middle);
    printf("%s %s. %s\n",First,Middle,Last);
}
```

6. Write a whole program that reads words from a file. Assume the file is named “words.txt”. Assume the file is in the current directory. Assume there is one word per line in the file. Count the number of times that the word “red” appears in the file. Don’t count things like “Red” or “RED”. Count lower case only. (10 points)

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

void main()
{
    int Count;
    char Word[20];
    FILE *InFile;

    InFile = fopen("words.txt","r");
    if (InFile == NULL)
    {
        printf("Can't open words.txt\n");
        exit(100);
    }
    Count = 0;
    scanf("%s",Word);
    while (!feof(InFile))
    {
        if (strcmp(Word,"red") == 0)
        {
            Count++;
        }
        scanf("%s",Word);
    }
    printf("Red Count = %d\n",Count);
}
```


3. Assume that you are creating a singly linked-list using the following data structure. The singly linked list will be a **stack**, using the variables given below. In the spaces given below, show how to allocate a new structure and link it into the **stack**. (10 Points)

```
// definitions
typedef struct abc
{
    struct abc *Next;
    int First;
    int Second;
    int Third;
}
ABC;

ABC *Head = NULL, *Temp;
int i,j,k;

int main(int argc,char *argv[])
{
    ...
    // Assume that the following code is part of a loop
    // read three integers
    scanf("%d %d %d",&i,&j,&k);
    // Allocate a new structure here and copy i, j, and k into it

    // Link the structure into the stack here

    ...
}
```

4. Assume that you are creating a singly linked-list using the following data structure. The singly linked list will be a **queue**, using the variables given below. In the spaces given below, show how to allocate a new structure and link it into the **queue**. (10 Points)

```
// definitions
typedef struct abc
{
    struct abc *Next;
    int First;
    int Second;
    int Third;
}
ABC;

ABC *Head = NULL, *Tail=NULL, *Temp;
int i,j,k;

int main(int argc,char *argv[])
{
    ...
    // Assume the following code is part of a loop
    // read three integers
    scanf("%d %d %d",&i,&j,&k);
    // Allocate a new structure here and copy i, j, and k into it

    // Link the structure into the queue here

    ...
}
```

5. You have created a singly linked list using the following definitions. Write a loop that frees all the storage for the linked list. (10 Points)

```
typedef struct ABC
{
    struct ABC *Next;
    char *First;
    char *Second;
}
ABC;
```

```
ABC *Head=NULL, *Tail=NULL, *Temp, *Temp2;
```

```
int main(int argc,char *argv[])
{
    ...
```

```
    // write your loop here
```

```
}
```

6. You have created a singly linked list using the following definitions. You now wish to delete the first item from the list that has a **Value** of 10. Show the loop that would be used to **FIND THE ITEM**. **DO NOT** delete the item. (10 Points)

```
typedef struct ABC
{
    struct ABC *Next;
    int Value;
}
ABC;

ABC *Head=NULL,*Tail=NULL,*Prev, *Temp;

int main(int argc, char *argv[])
{
    ...

    // Write your loop here

    ...
}
```

7. Using the same structures as for the previous problem, write a loop to add 10 to the Value component of each structure in the list. (10 Points)

```
int main(int argc, char *argv[])
{
    ...

    // Write your loop here

    ...
}
```

8. Using the following array definitions, write a loop that adds 10 to each element of the array. USE A POINTER to access the array elements. (10 Points)

```
int main(int argc,char *argv[])
{
    int Abc[20];
    int Size=20;
    int *Temp;
    int i;

    ...

    // Write Your Loop Here

}
```

9. Using the following definitions, write a segment of code that opens a file named “abc.txt” reads every character of the file (getc function) and displays the characters on standard out. Close the file when you are done. Exit the program if the file doesn’t exist. (10 Points)

```
int main(int argc,char *argv[])
{
    FILE *MyFile;
    char tc;

    ...

    // Write Your Code Here

}
```

10. Add **#define** statements to the beginning of this program, so that the rest of the code will compile properly. Define things in a way that makes the program read correctly. (10 Points)

```
// Add your #define statements here
```

```
int main(int argc, char *argv[])
{
    int i,j,k;

    ...
    i += TWO;
    j *= FIVE;
    k = j / TEN;
}
```

Extra Credit: Go back to problem 6, and examine the data structures. In the space below, write code to verify that the item has been found, and delete it from the list. Note: the curve will be computed based on the scores on the previous 10 problems. Extra credit will be added **AFTER** the curve is computed. (10 points)

NAME _____

1. Suppose you want to read a collection of strings from a file and store those strings in a **singly** linked list. You don't know how long the strings will be. Some will be short, and some will be very long. You want to store each string using as little storage as possible. Show the data structure that would be used to create this linked list. (10 Points)

```
typedef struct abc
{
    struct abc *Next;
    char *Value;
} ABC;
```

or

```
struct abc
{
    struct abc *Next;
    char *Value;
};
```

2. Assume that you want to do the same thing as in problem 1, but now you want to create a **doubly** linked list. Show the data structure that would be used to create a doubly linked list for the same problem. (10 Points)

```
typedef struct abc
{
    struct abc *Next;
    struct abc *Prev;
    char *Value;
} ABC;
```

or

```
struct abc
{
    struct abc *Next;
    struct abc *Prev;
    char *Value;
};
```

3. Assume that you are creating a singly linked-list using the following data structure. The singly linked list will be a **stack**, using the variables given below. In the spaces given below, show how to allocate a new structure and link it into the **stack**. (10 Points)

```
// definitions
typedef struct abc
{
    struct abc *Next;
    int First;
    int Second;
    int Third;
}
ABC;

ABC *Head = NULL, *Temp;
int i,j,k;

int main(int argc,char *argv[])
{
    ...
    // Assume that the following code is part of a loop
    // read three integers
    scanf("%d %d %d",&i,&j,&k);
    // Allocate a new structure here and copy i, j, and k into it

    Temp = (ABC *)malloc(sizeof(ABC));
    if (Temp == NULL)
    {
        exit(100); // possibly with error message
    }
    Temp->First = i;
    Temp->Second = j;
    Temp->Third = k;

    // Link the structure into the stack here

    Temp->Next = Head;
    Head = Temp;

    ...
}
```

4. Assume that you are creating a singly linked-list using the following data structure. The singly linked list will be a **queue**, using the variables given below. In the spaces given below, show how to allocate a new structure and link it into the **queue**. (10 Points)

```
// definitions
typedef struct abc
{
    struct abc *Next;
    int First;
    int Second;
    int Third;
}
ABC;

ABC *Head = NULL, *Tail=NULL, *Temp;
int i,j,k;

int main(int argc,char *argv[])
{
    ...
    // Assume the following code is part of a loop
    // read three integers
    scanf("%d %d %d",&i,&j,&k);
    // Allocate a new structure here and copy i, j, and k into it
```

Same as for problem 3.

```
    // Link the structure into the queue here

if (Head == NULL)
{
    Head = Temp;
}
else
{
    Tail->Next = Temp;
}
Temp->Next = NULL;
Tail = Temp;

    ...
}
```

5. You have created a singly linked list using the following definitions. Write a loop that frees all the storage for the linked list. (10 Points)

```
typedef struct ABC
{
    struct ABC *Next;
    char *First;
    char *Second;
}
ABC;
```

```
ABC *Head=NULL, *Tail=NULL, *Temp, *Temp2;
```

```
int main(int argc,char *argv[])
{
    ...
```

```
    // write your loop here
```

```
    for (Temp = Head ; Temp ; Temp = Temp2)
    {
        Temp2 = Temp->Next;
        free(Temp->First); // assuming dynamic allocation of strings
        free(Temp->Second);
        free(Temp);
    }
```

```
}
```

6. You have created a singly linked list using the following definitions. You now wish to delete the first item from the list that has a **Value** of 10. Show the loop that would be used to **FIND THE ITEM**. **DO NOT** delete the item. (10 Points)

```
typedef struct ABC
{
    struct ABC *Next;
    int Value;
}
ABC;

ABC *Head=NULL,*Tail=NULL,*Prev, *Temp;

int main(int argc, char *argv[])
{
    ...

    // Write your loop here

for (Temp = Head,Prev=NULL;
Temp && Temp->Value != 10;
Prev=Temp,Temp=Temp->Next);

    ...
}
```

7. Using the same structures as for the previous problem, write a loop to add 10 to the Value component of each structure in the list. (10 Points)

```
int main(int argc, char *argv[])
{
    ...

    // Write your loop here

for (Temp = Head ; Temp ; Temp=Temp->Next)
{
    *Temp += 10;
}

    ...
}
```

8. Using the following array definitions, write a loop that adds 10 to each element of the array. USE A POINTER to access the array elements. (10 Points)

```
int main(int argc,char *argv[])
{
    int Abc[20];
    int Size=20;
    int *Temp;
    int i;

    ...

    // Write Your Loop Here

for (i=0,Temp=Abc ; i<20 ; I++,Temp++)
{
    *Temp += 10;
}

}
```

9. Using the following definitions, write a segment of code that opens a file named "abc.txt" reads every character of the file (getc function) and displays the characters on standard out. Close the file when you are done. Exit the program if the file doesn't exist. (10 Points)

```
int main(int argc,char *argv[])
{
    FILE *MyFile;
    char tc;

    ...
    // Write Your Code Here
MyFile = fopen("abc.txt","r");
if (MyFile == NULL)
{
    exit(100); // perhaps with error message
}
tc = getc(MyFile);
while (!feof(MyFile))
{
    putc(tc,MyFile);
    tc = getc(MyFile);
}
fclose(MyFile);
}
```

10. Add **#define** statements to the beginning of this program, so that the rest of the code will compile properly. Define things in a way that makes the program read correctly. (10 Points)

```
// Add your #define statements here
```

```
#define TWO 2  
#define FIVE 5  
#define TEN 10
```

```
int main(int argc, char *argv[])  
{  
    int i,j,k;  
  
    ...  
    i += TWO;  
    j *= FIVE;  
    k = j / TEN;  
}
```

- Extra Credit:** Go back to problem 6, and examine the data structures. In the space below, write code to verify that the item has been found, and delete it from the list. Note: the curve will be computed based on the scores on the previous 10 problems. Extra credit will be added AFTER the curve is computed. (10 points)

```
if (Temp != NULL)  
{  
    if (Prev == NULL)  
    {  
        Head = Temp->Next;  
    }  
    else  
    {  
        Temp->Prev = Temp->Next;  
    } // This is enough for full credit  
    if (Temp->Next == NULL)  
    {  
        Tail = Prev; // many other correct solutions  
    }  
} // not necessary to free Temp
```