cs120s20-a.sg

Final Exam

Review of attempt 1

Finish review

Started on	Tuesday, 28 April 2020, 02:00 PM
Completed on	Tuesday, 28 April 2020, 03:39 PM
Time taken	1 hour 39 mins
Marks	70/100
Grade	**7** out of a maximum of 10 (**70**%)

1

Marks: 10

Study the following code below, assume all necessary includes are present:

```
char buffer[32] = { 0 };
char data[] = "EvggtEfwjdf";
int length = strlen(data);
char* cPtr = data; char* target = buffer;

int n = (length + 7) / 8;
switch (length % 8)
{
    case 0:
    do {
     *target++ = (*cPtr++) - 1;
    case 7: *target++ = (*cPtr++) - 1;
    case 6: *target++ = (*cPtr++) - 1;
    case 5: *target++ = (*cPtr++) - 1;
    case 4: *target++ = (*cPtr++) - 1;
    case 3: *target++ = (*cPtr++) - 1;
    case 2: *target++ = (*cPtr++) - 1;
    case 1: *target++ = (*cPtr++) - 1;
    } while (--n > 0);
}
printf("%s", buffer);
```

What is the **exact** output of the code above?

Answer:

DuffsDevice ✓

Correct

Correct

Marks for this submission: 10/10.

2

Marks: 10

Study the following code below, assume all necessary includes are present:

```
char buffer[32] = { 0 };
char data[] = "EvggtEfwjdf";
int length = strlen(data);
char* cPtr = data; char* target = buffer;

int n = (length + 7) / 8;
switch (length % 8)
{
     case 0:
     do {
      *target++ = (*cPtr++) - 1;
     case 7: *target++ = (*cPtr++) - 1;
     case 6: *target++ = (*cPtr++) - 1;
     case 5: *target++ = (*cPtr++) - 1;
     case 4: *target++ = (*cPtr++) - 1;
     case 3: *target++ = (*cPtr++) - 1;
     case 2: *target++ = (*cPtr++) - 1;
     case 1: *target++ = (*cPtr++) - 1;
     } while (--n > 0);
}
printf("%s", buffer);
```

What is the reason for writing in such a manner?

Choose one answer.

- a. To save memory ✗
- b. For performance via loop unrolling ✓ — Correct! Loops incur the cost of post-increment and a check per loop. So reducing the number of loops will increase performance. That is loop unrolling - attempting to perform action in batches to reduce the number of loops required.
- c. To confuse others trying to take your code ✗
- d. To lengthen the code to claim more work done ✗

Correct

Marks for this submission: 10/10.

3

Marks: 10

Given the files below...

Singleton.h

```
#ifndef SINGLETON_H
```

```
#define SINGLETON_H

struct SingletonData
{
    int Value;
};

struct SingletonData* GetSingleton();
void DestroySingleton();

#endif
```

Singleton.c

```
#include "Singleton.h"
#include <stdlib.h>

static struct SingletonData* gInstance = 0;

struct SingletonData* GetSingleton()
{
    if (gInstance == NULL)
        gInstance = (struct
        SingletonData*)malloc(sizeof(struct SingletonData));

    return gInstance;
}

void DestroySingleton()
{
    if (gInstance != NULL)
    {
        free(gInstance);
        gInstance = NULL;
    }
}
```

Assuming Singleton.h is included in main.c, what is the result of the line below if added to the main function:

```
gInstance = 0;
```

Choose one answer.

a. gInstance now has the value 0 ✗	Incorrect, it will not compile. main.c does not have access to the gInstance variable that is file-scoped in Singleton.c.
b. Run-time Crash ✗	
c. Compile Failure ✓	
d. Undefined Behavior ✗	

Incorrect

Marks for this submission: 0/10.

4

Marks: 10

Given the files below...

Singleton.h

```
#ifndef SINGLETON_H
#define SINGLETON_H

struct SingletonData
{
    int Value;
};

struct SingletonData* GetSingleton();
void DestroySingleton();

#endif
```

Singleton.c

```
#include "Singleton.h"
#include <stdlib.h>

static struct SingletonData* gInstance = 0;

struct SingletonData* GetSingleton()
{
    if (gInstance == NULL)
        gInstance = (struct
        SingletonData*)malloc(sizeof(struct SingletonData));

    return gInstance;
}

void DestroySingleton()
{
    if (gInstance != NULL)
    {
        free(gInstance);
        gInstance = NULL;
    }
}
```

What is the purpose of the code above?

Choose one answer.

- a. To make a function a variable ✗
- b. To copy the global variable ✗
- c. To confuse other users by writing more code ✗
- d. Provide access to a hidden global variable while ensuring only 1

Correct! That is the purpose of the singleton design pattern. Usually you'd

copy can exist at all times ✓	use it for global managers that should only have 1 instance at all times.

Correct
Marks for this submission: 10/10.

5

Marks: 10

Given a link-list of struct Node as shown below:

```
struct Node
{
    int value;
    struct Node* next;
};
```

Assume a proper AddNode function has been implemented.

Given the code below in main.c and all necessary includes are present:

```
struct Node* list = NULL;
int i = 0;

AddNode(&list);
AddNode(&list);
AddNode(&list);

i = sizeof(list);
printf("%i\n", i);
```

What is the **exact** printout in a 64-bit environment?

Answer:

8 ✓

Correct!

Correct
Marks for this submission: 10/10.

6

Marks: 10

Given a link-list of struct Node as shown below:

```
struct Node
{
    int value;
    struct Node* next;
};
```

Assume a proper AddNode function has been implemented.

Given the code below in main.c and all necessary includes are present:

```
struct Node* list = NULL;
int i = 0;

AddNode(&list);
```

```
AddNode(&list);
AddNode(&list);

i = sizeof(list);
printf("%i\n", i);
```

When is `sizeof(list)` resolved?

Choose one answer.

- a. Compile-time ✓ — Correct! The compiler will figure out the size by inferring the type of the variable or directly from a type provided. It then replaces the operator with the size value deciphered.
- b. Load-time (When the program loads up) ✗
- c. Run-time at the start of the function ✗
- d. Run-time at the line itself ✗

Correct

Marks for this submission: 10/10.

7

Marks: 10

Given the code below:

```
void Foo(const char* str)
{
 printf("%s\n", str);
}

int main(void)
{
 void* funcPtr = (void*)Foo;
 *funcPtr;
 return 0;
}
```

What is the result of this program?

Choose one answer.

- a. Unexpected behavior ✗
- b. Foo is invoked ✗
- c. Nothing happens ✗ — Incorrect. It will lead to a compile-time error as you cannot dereference a void pointer.
- d. Compile-time Error ✓

Incorrect
Marks for this submission: 0/10.

8

Marks: 10

Given the code below:

```
void Foo(const char* str)
{
 printf("%s\n", str);
}

int main(void)
{
 void* funcPtr = (void*)Foo;
 void(*noInputPtr)(void) = funcPtr;
 noInputPtr();
 return 0;
}
```

What is the result of this program?

Choose one answer.

- a. Nothing happens ✗
- b. Undefined Behavior ✓
- c. Compile-time Error ✗ — Incorrect. Behavior is undefined as the program will still attempt to invoke the function pointer but without the required arguments. As such, the function will access unknown memory that is meant for the arguments which gives unexpected results.
- d. Foo is invoked ✗

Incorrect
Marks for this submission: 0/10.

9

Marks: 10

Given the code below, assume all necessary includes are present:

```
int main(void)
{
 int i = 4407873;
 char* str = (char*)&i;
 printf("%s", str);
return 0;
}
```

What is the **exact** output of this program?

Answer:

ABC

Correct!

Correct
Marks for this submission: 10/10.

10

Marks: 10

Given the code below, assume all necessary includes are present:

```
int main(void)
{
 int i = ???;
 char* str = (char*)&i;
 printf("%s", str);
return 0;
}
```

What should be `???` to get an output of "FFF"?

Answer:

4605510

Correct!

Correct
Marks for this submission: 10/10.

Finish review

You are logged in as GOH Wei Zhe (Logout)

cs120s20-a

Validate HTML Section 508 Check WCAG 1 (2,3) Check