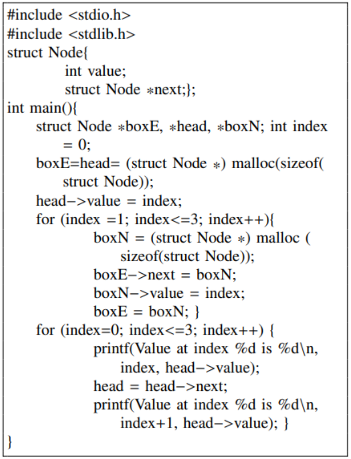
EE4013 : Assignment-1

By Ganraj Borade

October 4, 2021

Question

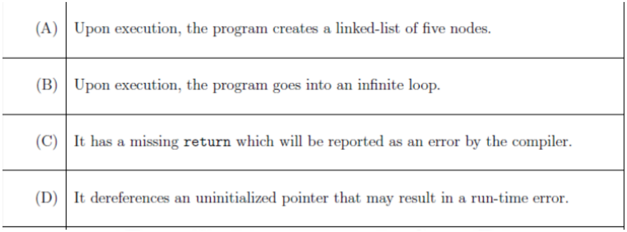
Q.35 Consider the following ANSI C Program.



```
#include <stdio.h>
#include <stdlib.h>
struct Node{
        int value;
        struct Node *next;};
int main(){
    struct Node *boxE, *head, *boxN; int index
        = 0;
    boxE=head= (struct Node *) malloc(sizeof(
        struct Node));
    head->value = index;
    for (index =1; index<=3; index++){
            boxN = (struct Node *) malloc (
                sizeof(struct Node));
            boxE->next = boxN;
            boxN->value = index;
            boxE = boxN; }
    for (index=0; index<=3; index++) {
            printf(Value at index %d is %d\n,
                index, head->value);
            head = head->next;
            printf(Value at index %d is %d\n,
                index+1, head->value); }
}
```

Which one of the following statements below is correct about the program?

Solution



(A)	Upon execution, the program creates a linked-list of five nodes.
(B)	Upon execution, the program goes into an infinite loop.
(C)	It has a missing `return` which will be reported as an error by the compiler.
(D)	It dereferences an uninitialized pointer that may result in a run-time error.

Answer : It dereferences an uninitialized pointer that may result in a run-time error

Explanation

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

The above structure is for linked list node. Let's now look at the main function in the given code :

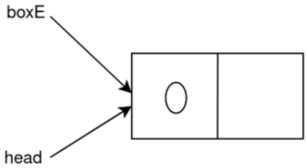
```
struct Node *boxE, *head, *boxN; int index = 0;
```

i.e there are 3 pointers : boxE, head and boxN. Now since these pointer variables are not initialized to null so they will point to some unwanted memory location. And we also have variable 'index'=0

Continuing

```
boxE=head= (struct Node *) malloc(sizeof(struct
    Node));
head->value = index;
```

Now we are going to assign boxE equal to head and then we are allocating the memory of size struct Node. So here a dynamic memory allocation will happen and a node will be created in a heap area.

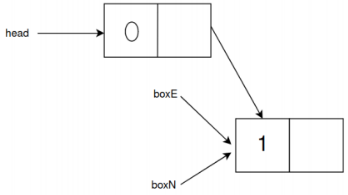


i.e boxE and head will be pointing to this node created and in this node for now garbage value is present. Since we are doing head $\rightarrow$ value $=$ index so, data value for this head is 0 as index is 0 now. And the next pointer will point to garbage location.

Continuing

```
for (index =1; index<=3; index++){
        boxN = (struct Node *) malloc (
            sizeof(struct Node));
        boxE->next = boxN;
        boxN->value = index;
        boxE = boxN; }
```

1. In the above for loop, initially index=1. The output of 1st iteration is as follows :

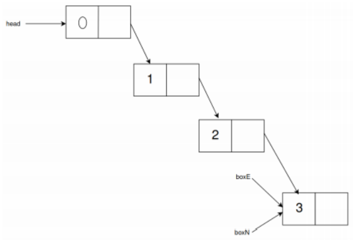


Continuing

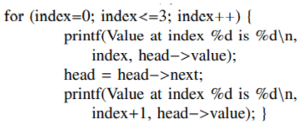
Because a new node boxN will be created as the loop runs. And since we are doing boxE$\rightarrow$next = boxN so as shown in the above figure, the next of boxE will point to boxN. After that since boxN$\rightarrow$value = index, so boxN value will be equal to 1 and after that since boxE= boxN, so again boxE will be pointing to boxN.

Continuing

2. Similarly after 3 iterations, we will get following output in the node form :



Continuing



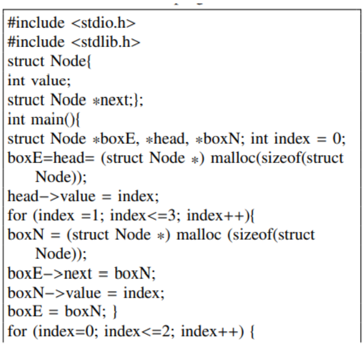
```
for (index=0; index<=3; index++) {
          printf(Value at index %d is %d\n,
              index, head−>value);
          head = head−>next;
          printf(Value at index %d is %d\n,
              index+1, head−>value); }
```

Now we can easily look at the output now. When index=0, then we will get output as :
Value at index 0 is 0 Value at index 1 is 1
Because in the second line of above code (head = head$\rightarrow$next), we will get head pointed to value 1.
Similarly at index=1, we will get : Value at index 1 is 1 Value at index 2 is 2
Similarly at index=2, we will get : Value at index 2 is 2 Value at index 3 is 3
Now when index=3, firstly we will get : Value at index 3 is 3

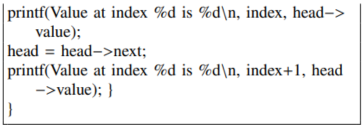
Continuing

Now head = head$\rightarrow$next, but now, head$\rightarrow$next is some unwanted pointer so it will contain some pointer to some garbage location therfore when head$\rightarrow$value will run, we could get a segmentation fault here because we are accessing an unwanted address.So, this program will be abnormally terminated.
Hence the answer is "It dereferences an uninitialized pointer that may result in a run-time error"

Correct Code



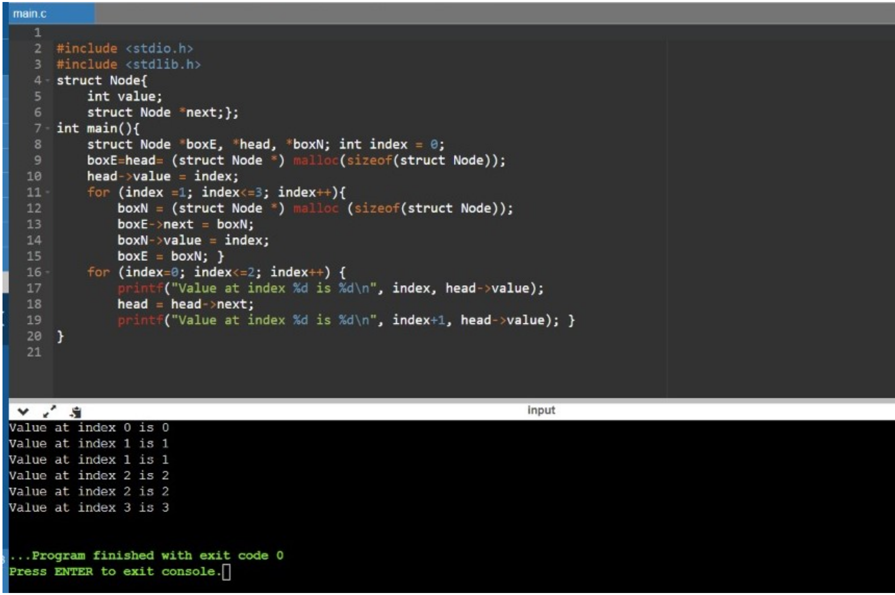
```
#include <stdio.h>
#include <stdlib.h>
struct Node{
int value;
struct Node *next;};
int main(){
struct Node *boxE, *head, *boxN; int index = 0;
boxE=head= (struct Node *) malloc(sizeof(struct
     Node));
head−>value = index;
for (index =1; index<=3; index++){
boxN = (struct Node *) malloc (sizeof(struct
     Node));
boxE−>next = boxN;
boxN−>value = index;
boxE = boxN; }
for (index=0; index<=2; index++) {
```



```
printf(Value at index %d is %d\n, index, head−>
     value);
head = head−>next;
printf(Value at index %d is %d\n, index+1, head
     −>value); }
}
```

That is in the last for loop just reduce last value of index to 2

Correct Code

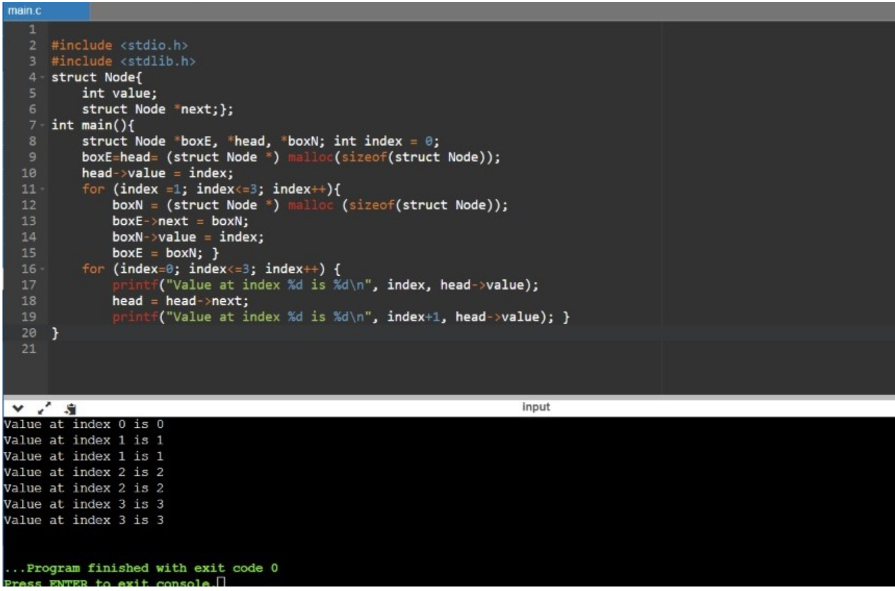

main.c

```c

#include <stdio.h>
#include <stdlib.h>
struct Node{
     int value;
     struct Node *next;};
int main(){
     struct Node *boxE, *head, *boxN; int index = 0;
     boxE=head= (struct Node *) malloc(sizeof(struct Node));
     head->value = index;
     for (index =1; index<=3; index++){
         boxN = (struct Node *) malloc (sizeof(struct Node));
         boxE->next = boxN;
         boxN->value = index;
         boxE = boxN; }
     for (index=0; index<=2; index++) {
         printf("Value at index %d is %d\n", index, head->value);
         head = head->next;
         printf("Value at index %d is %d\n", index+1, head->value); }
}

```

```
Value at index 0 is 0
Value at index 1 is 1
Value at index 1 is 1
Value at index 2 is 2
Value at index 2 is 2
Value at index 3 is 3


...Program finished with exit code 0
Press ENTER to exit console.
```

Incorrect Code

Here, the output is not printed according to expectation.



```c

#include <stdio.h>
#include <stdlib.h>
struct Node{
    int value;
    struct Node *next;};
int main(){
    struct Node *boxE, *head, *boxN; int index = 0;
    boxE=head= (struct Node *) malloc(sizeof(struct Node));
    head->value = index;
    for (index =1; index<=3; index++){
        boxN = (struct Node *) malloc (sizeof(struct Node));
        boxE->next = boxN;
        boxN->value = index;
        boxE = boxN; }
    for (index=0; index<=3; index++) {
        printf("Value at index %d is %d\n", index, head->value);
        head = head->next;
        printf("Value at index %d is %d\n", index+1, head->value); }
}

```

```
Value at index 0 is 0
Value at index 1 is 1
Value at index 1 is 1
Value at index 2 is 2
Value at index 2 is 2
Value at index 3 is 3
Value at index 3 is 3


...Program finished with exit code 0
Press ENTER to exit console.
```