LAB11 Part2-GDB Debugging

E17219
Nawarathna K.G.I.S.

1.Infinite_print

**The problem in this programme is that while loop executes an infinite loop.This is because the condition of the while loop never gets false.The condition is "difference>0".This never gets to 0(or to a negative number) because the variable 'difference' never gets updated inside the loop.Since the 'difference' variable is initialized when declared outside the loop , difference will always have the value 2 because it is the value of it when initialized.This bug can be fixed by updating the 'difference' each time we run through the loop.Following while loop will fix the problem.(The underlined statement is the change from the original.)

```
while(difference > 0)
{
   printf("1 more food needed");
   food++;
   difference = people - food;

}
```

**Note that standard return 0; statement is missing from the program as well.(That is not a bug)

2.two_node_list

**This program is buggy because of one reason.FreeTwoNodeList will release the allocated memory blocks to the system back.Therefore after calling this function there will be an error because allocated memory is already released.Therefore that function should be called at the last of the program.Following piece of code will give an idea about how to fix the bug.

```
int main()
{
        // Create a 2 node list
        Node *head = CreateTwoNodeList();

        // Add some values
        head-> value = 10;
        head->next->value = 20;

        // Add some values
        head-> value = 50;
        head->next->value = 80;
```

// Delete the list
FreeTwoNodeList(head);

}

<u>3.segmentation</u>
**There are few problems with this programme.

**First head node is allocated by static memory allocation.After that iterator gets head's address.Now the first thing to notice here is that head's value is not declared.Therefore it will take the garbage value.

**Now into the for loop.
==>In for loop there should be a dynamic memory allocation for newNodep.Following line should change as shown.

<u>Node* newNodep;</u> ⇒ **<u>Node* newNodep=(Node*)malloc(sizeof(Node));</u>**

**Just after the for loop.
==>Now the tail of the linked list should be assigned to NULL.Otherwise the end of the linked list cannot be identified.Therefore the following line should be included.

<u>iterator->next=NULL;</u>

==>After that the iterator should be assigned to the start of the linked list for printing values.But since heads value is a garbage value,the start of the linked list should be head.next. Therefore the following line should change as shown.

<u>iterator = &head; </u>=> **<u>iterator = head.next;</u>**

**Into the while loop.

==>Note that the iterator does not update its value and that will result in an infinite loop.To avoid that following line should be included inside the while loop.

<u>iterator=iterator->next;</u>

==>Furthermore the condition of the while loop should be changed as follows.Otherwise it will not print the last node's value.

<u>iterator->next != NULL</u> => **<u>iterator!= NULL</u>**

**After all of these intended behavior of the programme can be achieved.Fixed programme is shown below.

int main()

```
{
        Node head;
        Node* iterator = &head;
        // Create a linked list with 10 elements
        int i = 0;
        for(i = 0; i<10 ; i ++)
        {
                Node* newNodep=(Node*)malloc(sizeof(Node));
                iterator->next = newNodep;
                newNodep->value = i;
                iterator = newNodep;
        }

        iterator->next=NULL;

        // Printing the values
        iterator = head.next;
        while(iterator!= NULL)
        {
                printf("Value %d\n", iterator->value);
                iterator=iterator->next;
        }
}
```

4.pointers

**Everything about this programme is correct except a small mistake.
**A linked list with 3 nodes is created here.But the end of the linked list is not specified to NULL here.Therefore the following line should be included at the end of the programme.

nodep2->next=NULL;

**Since *nodep2* is the last node here,its' next should be specified to NULL.This should be done just before printing *"Linked list created\n"*.

5.linkedlist_lastvalue

**There are some problems with this program as well.Following line is a bug.

findLastNodeValue(node2);

**node2 is initialized as NULL.There is no memory allocation for node2.That means it is just a variable (pointer) and it does not have the next field.Therefore when the function call happens,because it does contain the next field , the programme will give an erroneous behaviour.

**Note that , node1->value is not initialized here.Therefore it will give an garbage value for the function call.

6.simpleProgram

**There is a small bug here in this programme.Note that the var2 is not used here.But in the for loop,when it iterates, *var2* value also increases.Therefore if we wanted to use it after the for loop it should be noted that *var2* is not 5 anymore.Therefore var2++ should be removed from the for loop iteration since it does not contribute to any function of the program.

**Other than that declaring variables that are unused in the program is memory wasting.Therefore since var2 variable should be removed to prevent memory wastage.

7.object

**Here in this programme ,addValues prints the values of var1,var2 and var3.But since they are local and not static after the function call,they vanishes from the memory.So to keep them alive those variables should be declared as static.Following change will give the expected behavior.

*int var1=10; => **static int var1 = 10;***
*int var2=20; => **static int var2 = 20;***
*int var3=30; => **static int var3 = 30;***