DEPARTMENT OF EECS
Indian Institute of Technology Bhilai
CS200 — SOFTWARE TOOLS AND TECHNOLOGIES Lab II
Scope: GDB & Valgrind
Difficulty Level: Moderate

Assignment 4
November 1, 2020

- Instructions
  - All answers will be in separate files in a single folder, named: `<group-id>_<group-name>`
  - Name files as `q<question-no>` without any extension. e.g., q2
  - Submit the `gdb` log files for questions that need `gdb`
  - Make a `tarball` for the folder that contains your answers
  - Compress the `tarball` using `gzip` before uploading on Piazza

1. Copy the following code snippet in a '.cpp' file and answer the questions mentioned below the code.

```
1. #include<iostream>
2. using namespace std;
3. int  func(int);
4. const int X = 5;

5. int main()
6. {
7.      cout << "Program is starting.\n";
8.      int  result;
9.      int  X  =  10;
10.     cout << "In the middle of the program.\n";
11.     result =  func(X);
12.     cout << "The result is " << result << ".\n";
13. }
14. int func(int y)
15. {
16.     int  answer;
17.     cout << "In the function.\n";
18.     answer = y;
19.     answer +=  X;
20.     int X =  20;
21.     cout << "In the  middle  of  the  function.\n";
22.     answer +=  X;
23.     cout << "The function is exiting.\n";
24.     return  answer;
25. }
```

(a) Using `gdb` commands record and report the values of the variables X, result, answer, and y at the following lines in the program: 7, 10, 12, 17, 21, 23 (do not use any additional `cout`; show the `gdb` commands used). For some of these lines, you may get an error because the variable doesnt exist within the scope of that line.

(b) On line 10, what is the value of result and where did it come from? If you run the program several times, does the value of result at line 10 ever change?

(c) What are the values printed for X on lines 10 and 17? Why are the printed values of X different?

2. Copy the following three code snippets in three '.c' files and debug them using `Valgrind` tool. In each case, show the errors and the way you correct them.

(a)
```
#include<stdlib.h>
#include<stdio.h>
#include<time.h>

const int SIZE = 1000;

int main() {
    int *iArray = malloc(sizeof(int) * SIZE);
    for (int i=0; i < SIZE; i++) {
        iArray[i] = i;
    }

    srand(time(NULL));
    int randNum = rand() % SIZE;

    printf("iArray[%d]: %d\n", randNum, iArray[randNum]);
    return 0;
}
```

(b)
```
#include <stdlib.h>
#include <stdint.h>

struct _List {
    int32_t* data;
    int32_t length;
};

typedef struct _List List;

List* resizeArray(List* array) {
    int32_t* dPtr = array->data;
    dPtr = realloc(dPtr, 15 * sizeof(int32_t));
    return array;
}
```

```
int main() {
    List* array = calloc(1, sizeof(List));
    array->data = calloc(10, sizeof(int32_t));
    array = resizeArray(array);

    free(array->data);
    free(array);
    return 0;
}
```

(c)
```
#include <stdlib.h>
#include <stdint.h>

int main() {
    char* dest = calloc(35, sizeof(char));
    char* source = malloc(34 * sizeof(char));

    for(int i = 0; i < 35; i++) {
        *(dest + i) = *(source + i);
    }

    return 0;
}
```

In this one, how many errors are found by `Valgrind`? Can you suppress the first error reported by `Valgrind` and show other(s)?

3. Copy the following code snippet in a '.c' file. Debug the code using `gdb` and/or `Valgrind` and show the steps/commands for detecting and rectifying the errors present in the code.

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

typedef struct {
    char *str;
    int len;
} CString;

CString *Init_CString(char *str)
{
    CString *p = malloc(sizeof(CString));
    p->len = strlen(str);
    strncpy(p->str, str, strlen(str) + 1);
    return p;
}
void Delete_CString(CString *p)
{
```

 Question 3 continues on the next page...

```
    free(p);
    free(p->str);
}

// Removes the last character of a CString and returns it.
char Chomp(CString *cstring)
{
    char lastchar = *( cstring->str + cstring->len);
    // Shorten the string by one
    *( cstring->str + cstring->len) = '0';
    cstring->len = strlen( cstring->str );
    return lastchar;
}

// Appends a char * to a CString
CString *Append_Chars_To_CString(CString *p, char *str)
{
    char *newstr = malloc(p->len + 1);
    p->len = p->len + strlen(str);
    // Create the new string to replace p->str
    snprintf(newstr, p->len, "%s%s", p->str, str);
    // Free old string and make CString point to the new string
    free(p->str);
    p->str = newstr;
    return p;
}

int main(void)
{
    CString *mystr;
    char c;
    mystr = Init_CString("Hello!");
    printf("Init:\n str: `%s' len: %d\n", mystr->str, mystr->len);
    c = Chomp(mystr);
    printf("Chomp '%c':\n str:`%s' len: %d\n", c, mystr->str, mystr->len);
    mystr = Append_Chars_To_CString(mystr, " world!");
    printf("Append:\n str: `%s' len: %d\n", mystr->str, mystr->len);
    Delete_CString(mystr);
    return 0;
}
```
