Name : Dev Adnani  
SID : 202212012

Subject : Systems Programming

Assignment : 3

1. Write c program 202212012\_Lab3\_1.c with:

fib() function generates the Fibonacci series of a given number n.

fact() function generates factorial of a given number n

numlen() function finds the length of a given number n

Using **array of function pointers (\*functions[])** call each of these functions respectively for a menu driven program. E.g. fib() is called when user selects 1, fact() for 2 and numlen() for 3, exit program when user enters 0

main() function implements menu and calls the respective functions as per menu selection by user

#include <stdio.h>

#include <stdlib.h>

void fib(int num)

{

int num1 = 0, num2 = 1, temp = 0;

if (num == 0)

{

printf("Fibonacci number: %d\n", num1);

return;

}

for (int i = 2; i <= num; i++)

{

temp = num1 + num2;

num1 = num2;

num2 = temp;

}

printf("Fibonacci number: %d\n", num2);

}

void fact(int num)

{

int fac = 1;

for (int i = 1; i <= num; i++)

{

fac \*= i;

}

printf("Factorial of %d is %d\n", num, fac);

}

void numlen(int n)

{

int count = 0;

while (n != 0)

{

n = n / 10;

count++;

}

printf("Length of a given number is %d\n", count);

}

int main()

{

int choice, fibo, fac, len;

void (\*opt[4])(int);

opt[0] = exit;

opt[1] = fib;

opt[2] = fact;

opt[3] = numlen;

while (1)

{

printf("\nSelect Your choice");

printf("\n1) Fibonacci");

printf("\n2) Factorial");

printf("\n3) Find length of a given number");

printf("\n4) Clear Screen");

printf("\n0) Exit");

printf("\nEnter your choice: ");

scanf("%d", &choice);

switch (choice)

{

case 1:

printf("Enter number to Get Fibonacci series: ");

scanf("%d", &fibo);

(\*opt[choice])(fibo);

break;

case 2:

printf("Enter number for factorial: ");

scanf("%d", &fac);

(\*opt[choice])(fac);

break;

case 3:

printf("Enter number to find length: ");

scanf("%d", &len);

(\*opt[choice])(len);

break;

case 4:

system("clear");

break;

case 0:

exit(0);

default:

printf("Invalid choice");

break;

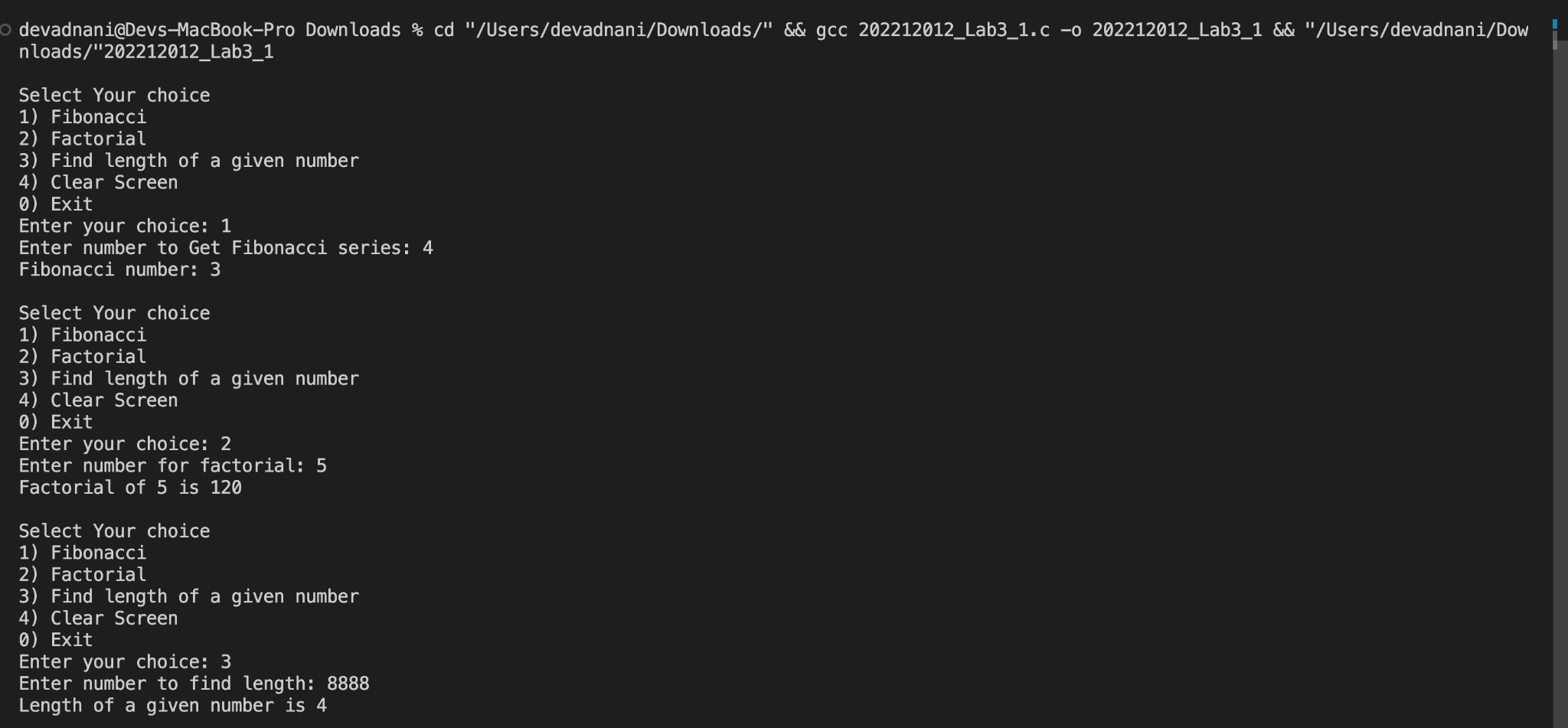
}

}

return 0;

}

Screenshot :



Q2 : Write a c program StudentID\_Lab3\_2.c which implements recursive function recursive\_strlen(char \*str) to find the length of the string recursively which is called from main function to get the length of the string.

Code :

#include <stdio.h>

int strlenRec(char \*str)

{

if (\*str == '\0')

return 0;

else

return 1 + strlenRec(str + 1);

}

int main()

{

char str[50];

printf("Enter a string: ");

scanf("%s",str);

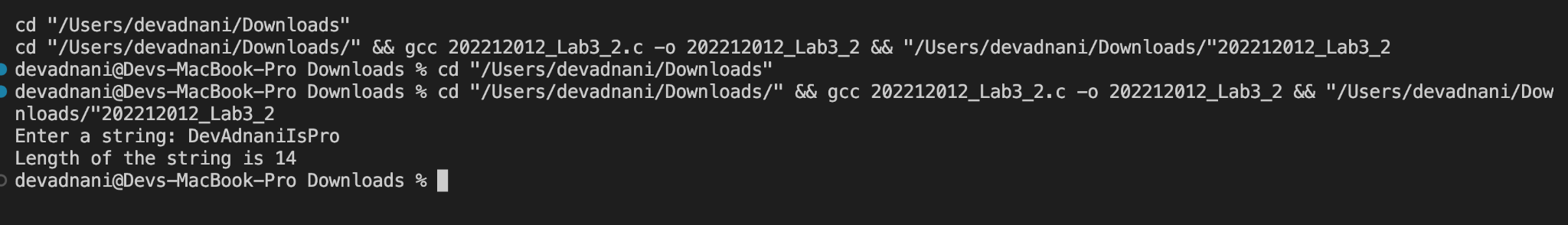
int length = strlenRec(str);

printf("Length of the string is %d\n",length);

return 0;

}

Screenshot :



Q3 : Write a c program StudentID\_Lab3\_3.c for this problem. You are given a task to keep track of all the programs running on a computer. You are required to implement a **doubly linked list** (non-circular) where in each node will store the following program information and will have pointer to the next and previous nodes. There will also be head and tail pointers pointing to first and last nodes respectively.

#include <stdio.h>

#include <stdint.h>

#include <stdlib.h>

#include <string.h>

#include <sys/time.h>

#define SUBMITTED 0

#define RUNNING 1

#define FINISHED 2

struct program

{

int prog\_id;

char prog\_name[50];

char uid\_executing[25];

uint64\_t start\_time;

uint64\_t time\_elapsed;

int prog\_status;

struct program \*next;

struct program \*prev;

} \*head = NULL, \*tail = NULL;

int progid = 1;

void addProcess()

{

char prog\_name[50], uid[50];

printf("Enter the program name: ");

scanf("%s", prog\_name);

printf("Enter the user id: ");

scanf("%s", uid);

struct program \*newNode;

newNode = (struct program \*)malloc(sizeof(struct program));

newNode->prog\_id = progid++;

strcpy(newNode->prog\_name, prog\_name);

strcpy(newNode->uid\_executing, uid);

newNode->prog\_status = SUBMITTED;

newNode->start\_time = 0;

newNode->time\_elapsed = 0;

if (head == NULL)

head = newNode;

if (tail != NULL)

{

tail->next = newNode;

newNode->prev = tail;

}

tail = newNode;

printf("New Program is Created!!\n");

}

void printAllProcesses()

{

struct program \*curr = head;

printf("\nList of all Programs:\n");

while (curr != NULL)

{

printf("Program ID:%d\tProgram Name:%s\tUserId:%s\tStartTime:%ld\tTimeElapsed:%ld\tStatus:", curr->prog\_id, curr->prog\_name, curr->uid\_executing, curr->start\_time, curr->time\_elapsed);

if (curr->prog\_status == SUBMITTED)

printf("Submitted\n");

else if (curr->prog\_status == RUNNING)

printf("Running\n");

else

printf("Finished\n");

curr = curr->next;

}

}

struct program \*searchProgram(int id)

{

struct program \*curr = head;

while (curr != NULL)

{

if (curr->prog\_id == id)

return curr;

curr = curr->next;

}

return curr;

}

uint64\_t getCurrentTime()

{

struct timeval tv;

if (gettimeofday(&tv, NULL) == 0)

return (uint64\_t)tv.tv\_sec;

else

return 0;

}

void StatusToRunning()

{

int pid;

printf("Enter Program ID:");

scanf("%d", &pid);

struct program \*prog = searchProgram(pid);

if (prog != NULL)

{

if (prog->prog\_status == SUBMITTED)

{

prog->prog\_status = RUNNING;

prog->start\_time = getCurrentTime();

printf("Process %d is now running!!\n", pid);

}

else if (prog->prog\_status == RUNNING)

printf("Process %d is already running!!\n", pid);

else

printf("Process %d is finished!!\n", pid);

}

else

printf("Process is not present!!\n");

}

void statusToFinished()

{

int pid;

printf("Enter Program ID:");

scanf("%d", &pid);

struct program \*prog = searchProgram(pid);

if (prog != NULL)

{

if (prog->prog\_status == SUBMITTED)

{

printf("Process %d is not running!!\n", pid);

}

else if (prog->prog\_status == RUNNING)

{

prog->prog\_status = FINISHED;

prog->time\_elapsed = (uint64\_t)getCurrentTime - (uint64\_t)prog->start\_time;

printf("Process %d is now finished!!\n", pid);

}

else

printf("Process %d is finished!!\n", pid);

}

else

printf("Process is not present!!\n");

}

void updateTimeElapsed()

{

struct program \*curr = head;

while (curr != NULL)

{

if (curr->prog\_status == RUNNING)

curr->time\_elapsed = (uint64\_t)getCurrentTime - (uint64\_t)curr->start\_time;

curr = curr->next;

}

}

void removeFinishedProcess()

{

struct program \*curr = head;

while (curr != NULL)

{

if (curr->prog\_status == FINISHED)

{

if (curr == head)

{

head = head->next;

}

else if (curr == tail)

{

tail = tail->prev;

tail->next = NULL;

}

else

{

curr->prev->next = curr->next;

curr->next->prev = curr->prev;

struct program \*temp = curr->next;

curr = temp;

}

}

curr = curr->next;

}

printf("Successfully Deleted!!\n");

}

void main()

{

void (\*functions[7])() = {addProcess, StatusToRunning, statusToFinished, updateTimeElapsed, removeFinishedProcess, printAllProcesses, exit};

int ch;

while (1)

{

printf("\n1) Add new program");

printf("\n2) Change status from submitted to running");

printf("\n3) Change status from running to completed");

printf("\n4) Update time-elapsed for all programs");

printf("\n5) Remove all the completed programs from the list");

printf("\n6) Print all the programs information");

printf("\n7) Exit");

printf("\nEnter your choice: ");

scanf("%d", &ch);

if (ch >= 1 && ch <= 7)

functions[ch - 1]();

else

printf("Invalid Option!!\n");

}

}

// Screenshot :

