PES UNIVERSITY

Electronic City Campus, Hosur road,
Bengaluru-560100, Karnataka.

PROJECT REPORT

on

"Rocket Launch Assist by Applying CERT principles to help in discovery of extraterrestrial Life"

for the Subject,

Secured Programming with C(*UE18CS257C*)

in

Bachelor of Engineering

from

Department of Computer Science and Engineering

For the Academic year 2019-2020

By

R .Harish ***Tejeshwar .U***

PES2201800541 ***PES2201800276***

Under the Guidance of

Dr. D.C. Kiran

Assistant Professor

Department of Computer Science and Engineering

PES University EC Campus.

PES UNIVERSITY

Electronic City Campus, Hosur road,
Bengaluru-560100, Karnataka.
Department of Computer Science and Engineering

CERTIFICATE

Certification that the project work entitled ***Rocket Launch Assist*** *carried out by* ***Tejeshwar.U (PES2201800276)*** *and* ***R.Harish(PES2201800541)*** *bona fide students of 4th semester in partial fulfillment of the award of bachelor of Engineering in* ***PES University*** *during the year 2022.*

This project report is approved as it satisfies the academic requirements in respect of the project work prescribed for the course ***Secure Programming with C.***

GUIDE	**HOD CSE**
Dr. DC KIRAN	**Dr. SANDESH**

DECLARATION

*We hereby declare that the project entitled **‘Rocket Launch Assist’** submitted for Bachelor of Computer Science Engineering is our original work and the project is not based on awards of any degree, associateship, fellowship or any other similar titles.*

Signature of Students:

R.Harish ***Tejeshwar.U***

Place: Bangalore

Date: 12-06-2020

TITLE

Rocket Launch Assist by Appling CERT Principles to help in the discovery of extra terrestrial life is a Project which allows user to find extra-terrestrial life using C and CERT rules and recomendations.

ABSTRACT

The application is intended for real-life rocket simulation/ rocket launch purposes where data integrity is the key requirement as well as reliability. The application strives for a reliable rocket launch management system and it walks through the process of selecting an appropriate launch vehicle, entering the payload to be deployed as well as the actual rocket launch.

Rocket countdown is simulated and every phase of the rocket launch is broken down such as the countdown phase, rocket reaching orbital velocity and escape velocity, launch vehicle detachment as 3 different phases as well as the satellite monitoring protocol for monitoring the data sent by satellite. The helpful combination of C and secure programming ensures a safe and reliable approach to monitor a rocket launch without any C based errors such as segmentation fault etc.

The main feature is that CERT recommendations are ensured thus making the rocket launch process a hassle free one. Several recommendations regarding strings arrays and abstraction have been subsumed maintaining high-reliability standards. C based errors maybe raised during data transmission, entry details-null termination and sanitization based errors, launch vehicle selection, during rocket countdown- initialization based errors. These have been mitigated by abiding CERT rules for safe and reliable green programming.

TABLE OF CONTENTS

1.0 INTRODUCTION

In today's world space missions are very essential in the fact that they help to unearth the mysteries that space beholds. Thus a mission that requires a satellite payload for exploration is a natural thought. Many such missions have taken places such as the Curiosity Rover mission, India's Chadraayan and much more. A general requirement is **safe communication protocols with the satellite** so that essential image transfer and data transfer can take place

What we want to demonstrate that C is up for challenges like this. With excellent computing speeds and low level hardware interactions along with excellent efficiency in embedded applications C would be a boon to the space research . Sadly C's very error prone nature makes C not an ideal choice for these applications. Because who would want a rocket to fail reach orbital velocity just because of a segmentation fault which totally evaded the programmer's eagle vision? Being a 'cost heavy' investment a single error can decide failure or success of a space launch. In such cases safety is the best requirement sadly C's philosophy dictates otherwise.

This is where 'Secure Programming' comes to the rescue. Using safe CERT principles we ensure that reliability is achieved in the communication protocol and also safety. Green programming with massive effects -> Now C can also be used for such endeavors without worry or discomfit. How small errors creep up and how secure programming saves the day is what our communication module revolves around. Every single error possible to has been shown in the application with how secure programming manages it as well.

All hail the deadly combo of C and Secure Programming!

We've kept the reader for too long. Let's delve into this 'futuristic mission' and why C the underdog is the best for it.

2.0 SYSTEM REQUIREMENTS

2.1 SOFTWARE REQUIREMENT SPECIFICATION

A moderately powered computer would be enough because we use C programming language. C ‘s dynamics with low level hardware makes it ideal to run even on embedded systems with ease.Secure programming requires a well-functioning string library because the in-situ C string library is not CERT-proof, so we have used the **managed string library** for entries.

The well-known gcc compliler is used with C version C99. Recent version like C 11 compiler would also work with the implementation done in the project.Colors have been added just to make the program more – human friendly by using rlutil header. Different stages of the mission **ROCKET LAUNCH** could be analyzed with various parameters, so the *rlutil header* has been used for glorifying the application.

Functions in C makes scaling easier. Also note that the rlutil header as optional and a minimalistic application without the colors can also be used.

When we try to use rlutil header file in multiple files during implementation of the project, global declarations in the rlutil header itself may cause problems, so necessary solutions have been implemented to alleviate this concern.

Since we use C programming language its capability and simplicity enables AI to detect and autocorrect few errors and also be able to guide the payload to do its operations when it is too far to be controlled.

2.2 SAFETY REQUIREMENT SPECIFICATION

2.2.1 SAFETY FEATURES INCLUDED

We use C programming language in our project implementation for its simplicity, but we need to consider the vulnerable situations we face during the implementation. C does give warnings during compilation with respect to pointers, string manipulation, expression evaluations, scope of a variable, function.etc and memory issues.

Scope: We need to take care of msleep function used for aesthetic flow of the application and thus b use rlutil header and we have switched to sleep function declared in stdlib header whenever necessary to prevent collisions.

Null Termination: This vulnerability generally occurs during String operations. During copying or concatenation of string, the final length of string may not have the space for null character, so we get the made sure to store maximum of a given length which is warned to the users. Thus, Null character is always added at the end.

Strings: Use of the managed string library – getstrptr_m and other managed function have been used so that string vulnerabilities are avoided.Null termination check has been ensured

Buffer overflow has been prevented because of the managed string library incorporation. Proper data sanitization has been implemented using the strspn function

Pointers: If any are NULL initialized and properly deallocated so that dangling pointers and dereferencing issues, segmentation faults caused by string allocation have been alleviated

Several other workarounds such as maximum size for string inputs fgets for character size limitation have all been followed and have been included in the declaration section.

Memory: Proper allocation & deallocation has been followed. Sensitive information previously stored is cleared and the memory is only then allocated.

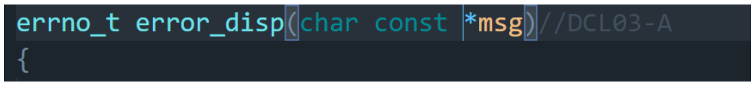
Stack allocations are of appropriate size to ensure that stack overflow(while user inputs) is abated

To prevent stack overflow *'fstack-protector-strong'* the gcc flag to ensure stack safety has been implemented.

2.2.2 SAFETY RULES AND REGULATIONS

Declarations:

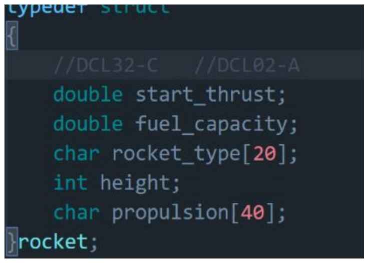
DCL03 -A Place const as rightmost specifier



```
errno_t error_disp(char const *msg)//DCL03-A
{
```

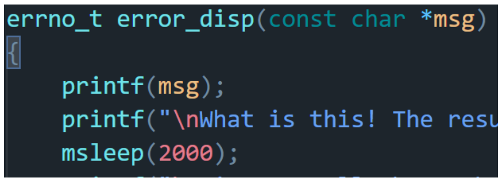
DCL32
Guarantee identifiers are unique

DCL02-A
Use visually distinct identifiers



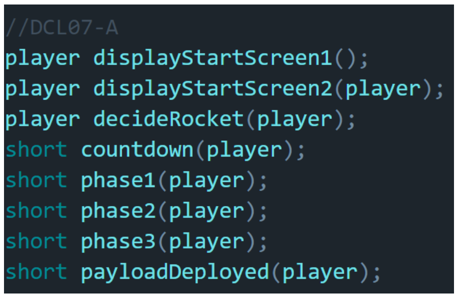
```
typedef struct
{
    //DCL32-C   //DCL02-A
    double start_thrust;
    double fuel_capacity;
    char rocket_type[20];
    int height;
    char propulsion[40];
}rocket;
```

DCL09 -A Functions that return an error with errno_t



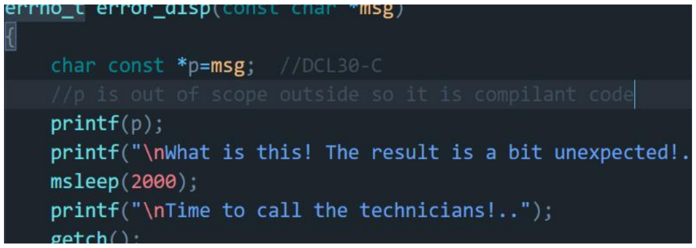
```
errno_t error_disp(const char *msg)
{
    printf(msg);
    printf("\nWhat is this! The resu
    msleep(2000);
```

DCL07-A Every function has a prototype



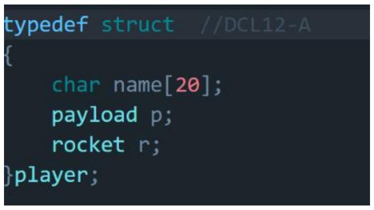
```
//DCL07-A
player displayStartScreen1();
player displayStartScreen2(player);
player decideRocket(player);
short countdown(player);
short phase1(player);
short phase2(player);
short phase3(player);
short payloadDeployed(player);
```

DCL30 -C   Declare objects with appropriate storage durations.



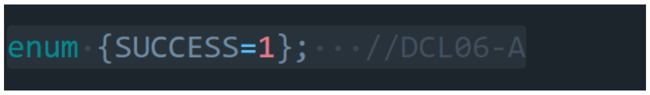
```
errno_t error_disp(const char *msg)
{
    char const *p=msg;  //DCL30-C
    //p is out of scope outside so it is compilant code
    printf(p);
    printf("\nWhat is this! The result is a bit unexpected!.
    msleep(2000);
    printf("\nTime to call the technicians!..");
    getch();
```

DCL12 -A Create and use abstract data types



```
typedef struct  //DCL12-A
{
    char name[20];
    payload p;
    rocket r;
}player;
```

DCL06-A – Use meaningful symbolic constants to represent values in program logic



```
enum {SUCCESS=1};   //DCL06-A
```

Integer and Floating Point:

FLP 32C - Ensure domain integrity in math functions

```
float getVelocity(player p)
{
   return sqrt(12.1*p.r.fuel_capacity/-p.r.start_thrust);
}
```

Expressions:

EXP 00A – Use parantheses for operator precedence

```
int capability_pass(player p)
{
   double x=p.r.fuel_capacity;
   double y=p.r.start_thrust;
   return x/(10*x)-y/(10*y) == 0 ? 1 :0;
}
```

EXP 03A – The size of a pointer does not determine the struct size

EXP 33C- Do not reference unitialized variables

```
short c;
while (c != -1)
{
   printf("%d", c);
   Sleep(1000);
   printf("\b");
   c--;
}
```

PRE 00A- Prefer inlines over macros

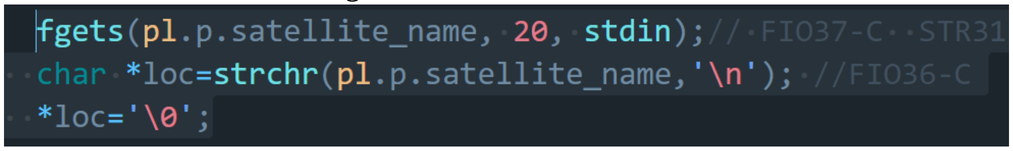
PRE 03A- Ensure that a library function is not called by mistake and thus undef it

PRE 07C- Avoid using repeated question marks – triggers trigraphs

```
char genomesequence[] = "gcauacac??'??]??)??(ucgcuaugucgauaacaac";
```

Files and Memory :

FIO 37C- Do not assume fgets reads all the characters data



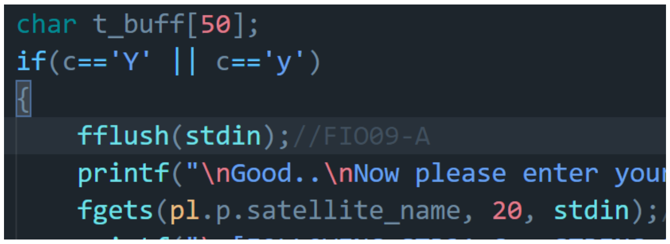
```
fgets(pl.p.satellite_name, 20, stdin);// FIO37-C  STR31
  char *loc=strchr(pl.p.satellite_name,'\n'); //FIO36-C
  *loc='\0';
```

MEM 02A- Do not cast the return value from malloc.

MEM 05A – Avoid large stack allocations

Utilize f-stack protector strong for stack overflow and buffer overflow defense mechanism

FIO09-A Use fflush when data integrity is important (Sensitive data protection)



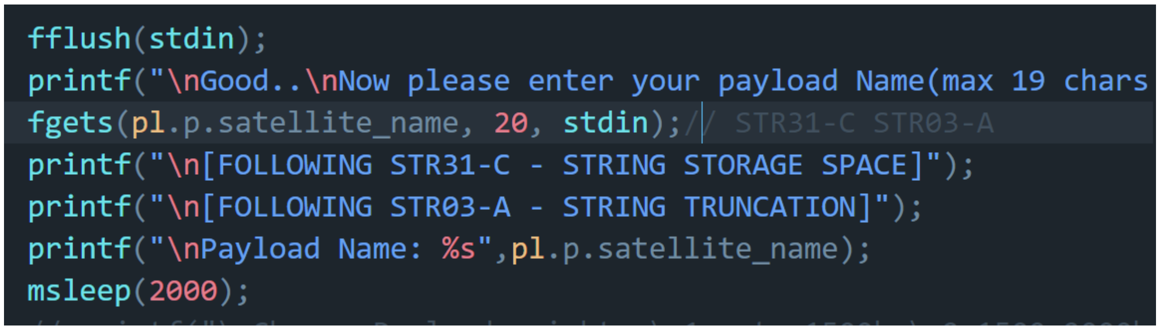
```
char t_buff[50];
if(c=='Y' || c=='y')
{
    fflush(stdin);//FIO09-A
    printf("\nGood..\nNow please enter your
    fgets(pl.p.satellite_name, 20, stdin);
```

Strings and Arrays :

ARR 30-C -Guarantee the array indices are within the specified range

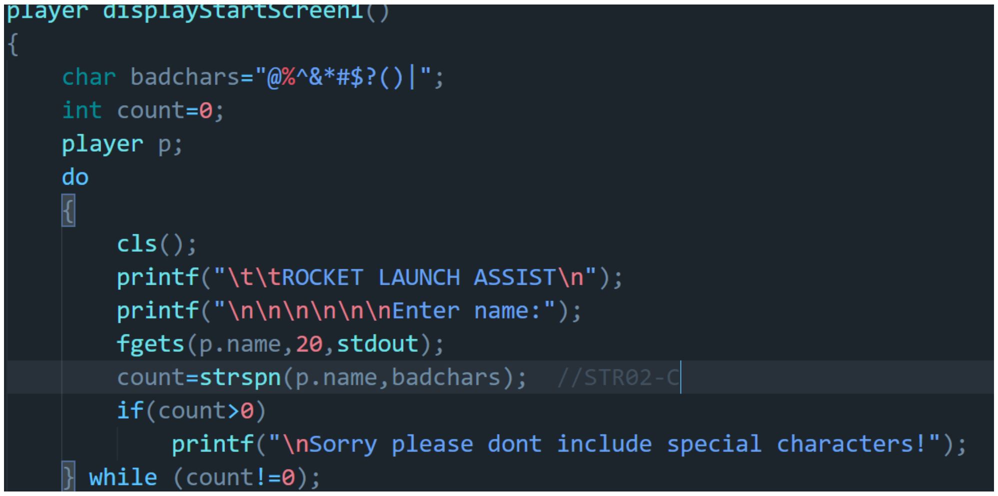
STR 03-A – Don’t truncate a null byte terminated string

STR 31-C – Ensure sufficient space for string allocation is guaranteed



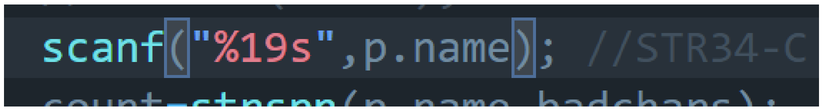
```
fflush(stdin);
printf("\nGood..\nNow please enter your payload Name(max 19 chars
fgets(pl.p.satellite_name, 20, stdin);// STR31-C STR03-A
printf("\n[FOLLOWING STR31-C - STRING STORAGE SPACE]");
printf("\n[FOLLOWING STR03-A - STRING TRUNCATION]");
printf("\nPayload Name: %s",pl.p.satellite_name);
msleep(2000);
```

STR 02C- Data sanitization



```
player displayStartScreen1()
{
    char badchars="@%^&*#$?()|";
    int count=0;
    player p;
    do
    {
        cls();
        printf("\t\tROCKET LAUNCH ASSIST\n");
        printf("\n\n\n\n\n\nEnter name:");
        fgets(p.name,20,stdout);
        count=strspn(p.name,badchars);  //STR02-C
        if(count>0)
            printf("\nSorry please dont include special characters!");
    } while (count!=0);
```

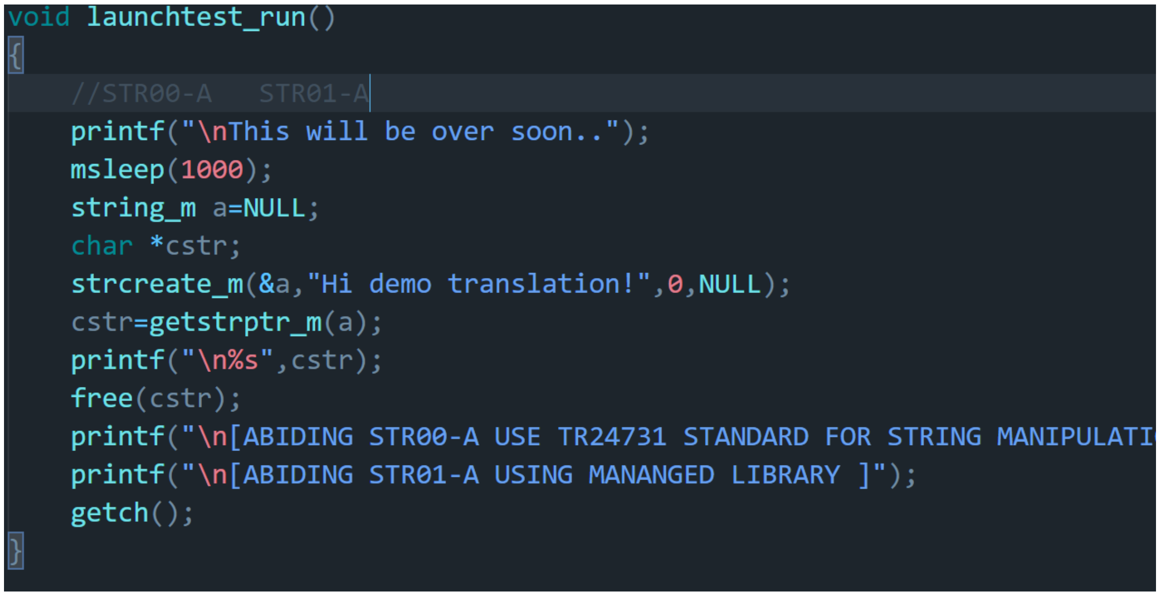
STR34-C
Don't copy data from unbounded source to fixed length array (The Buffer overflow)



```
  scanf("%19s",p.name); //STR34-C
  count=strspn(p.name,badchars);
```

STR 00A - Utilize the TR 24731 standard for data integrity and string operations

STR 01A - Use the managed string function library



```
void launchtest_run()
{
    //STR00-A   STR01-A
    printf("\nThis will be over soon..");
    msleep(1000);
    string_m a=NULL;
    char *cstr;
    strcreate_m(&a,"Hi demo translation!",0,NULL);
    cstr=getstrptr_m(a);
    printf("\n%s",cstr);
    free(cstr);
    printf("\n[ABIDING STR00-A USE TR24731 STANDARD FOR STRING MANIPULATI
    printf("\n[ABIDING STR01-A USING MANANGED LIBRARY ]");
    getch();
}
```

STR 05A - Prefer making string literals const qualified

3.0 SYSTEM DESIGN

3.1 MODULARITY

Modularity is about organizing the program into different subsections which enables us to code easier, provide scope for good scalable project, easy to understand and modify. It helps us to maintain remote phases of the project in different modules which enables easier modification and module testing.

Rocket Launch Assist project is implemented using five different modules namely main.c, gen_func.c, launch.c, payload.c, find_life.c.

main.c :

This is main file from where execution starts with main() function. It has main function calls to different stages in each phase of the launch. Phases of Rocket Launch are launch phase, payload phase, finding life phase.

gen_func.c

This file contains all functions used for general functionality inside the main functions of each phase. The functions slowprint – prints output with time delay to give onsite feel of rocket launch, line_displayer – which surrounds the heading for distinguishing between normal text, flashytext - erases the witten text to rewrite the altitude values in three department stages, dots – to show the payload is sending data, getDigits – which generates digits and displays them after using flashytext function.

launch.c

The displayStartScreen1 function is initially called from main.c file which gets controller name as input and checks for data sanitization using blacklisting technique. Then in displayStartScreen2 function gets payload name and takes input of rocket name in decideRocket function and its capabilities are assigned accordingly.

payload.c

Now countdown for launch starts in countdown function. Lower part, middle part and payload get detached in phase1, phase2, phase3 functions respectively.

find_life.c

Payload has been deployed in previous stage. Now it goes to Planet Omricon and searches for life in payloadDeployed function, where launchtest_run function is activated which inturn activates AI assist in our mission. The previous function finds genomes and sends data to the controller.

headers.h

It contains all the structures , typedefs and function prototypes used in our mission.

3.2 READABILTY

We made sure that program function names are readable so it is easier for future controller and programmers to modify in future. We tried to add as comments where we felt necessary. We used good variable names and in some cases we used typedefs for data types to give a meaningful names such as **typedef int errorno_t** and **enum {SUCCESS=1}.**

3.3 ABSTRACTION

Abstraction is an implementation of using an application without knowing its background details, mainly in module division and structure definition. We use gen_func.c file in all three stages of our mission where we call functions in this modules which act as an abstraction to all other three launch, payload and find_life files/stages. We get to select rocket name in the decideRocket function where it uses player structure to initialize its thrust, propulsion and fuel capacity as an abstraction. Necessary typedefs and enumerations have been used for structure abstractions. So this ensures abstraction and easier implementation and aesthetics.

4.0 IMPLEMENTATION

Rocket Launch Assist is implemented using different modules like main.c, gen_func.c, launch.c, payload.c, find_life.c, header.h.

main.c file is the place where execution starts. The functions in it are looked up in header file, thus executing user-defined function in other modules. gen_func.c file holds all the general functions used by primary functions in the project. launch.c holds implementation details of selecting project controller person, then deciding payload details such as name of the payload, start Thrust, propulsion, payload limit and fuel capacity. payload.c determines the countdown of the rocket, and different phases of detachment of rocket parts. In find_life.c the final payload goes to planet Omicron searching for life, gets genome sequence if life exists.

headers.h

```
#include<stdio.h>
#include<errno.h>
#include<string.h>
#include<stdlib.h>
// #include<graphics.h>
#include<windows.h>
#include<math.h>
// #include"rlutil.h"
#include "string_m/string_m.h"
// #include "string_m/string_m_internals.h"


extern int errno;
typedef int errno_t;


enum {SUCCESS=1};   //DCL06-A

typedef struct //DCL05-A
```

```
{
    char satellite_name[20];
    int weight;
}payload;

typedef struct
{
    //DCL32-C   //DCL02-A
    double start_thrust;
    double fuel_capacity;
    char rocket_type[20];
    int height;
    char propulsion[40];
}rocket;

typedef struct  //DCL12-A
{
    char name[20];
    payload p;
    rocket r;
}player;

//DCL07-A
void line_displayer(int color);
//Displays dashed line of a color(0-15)-WORKS ONLY ON SMALL SCREEN!

void slowprint(int color, char *text);
int capability_pass(player p);
float getVelocity(player p);
short getDigits(short num);

void flashy_text(int color, char *text);
```

```
/*
Takes a colorcode(ANSI/ DOS based from 0-15 only) and text to be displayed
Make sure youre sending codes from 0-7 ONLY
*/

void dots(int color, int n);
//Takes a color code(0-15) and number of dots to be displayed

player displayStartScreen1();
player displayStartScreen2(player);
player decideRocket(player);
short countdown(player);
short phase1(player);
short phase2(player);
short phase3(player);
short payloadDeployed(player);

errno_t error_disp(const char *);//DCL09-A
```

main.c

```
#include "headers.h"

int main()
{
    player p=displayStartScreen1();
    p=displayStartScreen2(p);
    p=decideRocket(p);
    countdown(p);
    phase1(p);
    phase2(p);
    phase3(p);
    payloadDeployed(p);
}
```

gen_func.c

```
#include"headers.h"

//PRE03-A
#if error_disp
    #undef error_disp
#endif


void slowprint(int color, char *text)
{
   //setColor(color);
    for (int i = 0; i < strlen(text); i++)
    {
        putchar(text[i]);
        Sleep(90);
    }
    // resetColor();
}

void line_displayer(int color)
{
   //setColor(color);
```

```
    for (size_t i = 0; i < 117; i++)
    {
        printf("-");
    }
    printf("\n");
}

void flashy_text(int color, char *text)
{
    int count;
    for (size_t j = 0; j < 2; j++)
    {
        if (j == 0)
        {
           //setColor(color);
            count = printf("%s", text);
           Sleep(1000);
        }
        else
        {
            for (size_t i = 0; i < count; i++)
                printf("\b");
           //setColor(color);
            count = printf("%s", text);
           Sleep(1000);
        }
        for (size_t i = 0; i < count; i++)
            printf("\b");
       //setColor((color + 8)%16);
        printf("%s", text);
       Sleep(1000);
    }
    // resetColor();
    //printf("\n");       //Sets cursor to newline after flashing ADD if required!
}

void dots(int color, int n)
{
   //setColor(color);
    for (size_t i = 1; i <= n; i++)
    {
        printf(".");
       Sleep(500);
    }
   //setColor(0);
   Sleep(500);
    int count = 0;
    for (size_t i = 1; i <= n; i++)
    {
        if (count == 1)
            for (size_t j = 1; j <= 2; j++)
                printf("\b");
        else
        {
            printf("\b");
            count++;
        }
        printf(".");
       Sleep(500);
    }
}

int capability_pass(player p)
{
    double x=p.r.fuel_capacity;
    double y=p.r.start_thrust;
    return x/(10*x)-y/(10*y) == 0 ? 1 :0;
}

float getVelocity(player p)
{
    return sqrt(12.1*p.r.fuel_capacity/-p.r.start_thrust);
}

errno_t error_disp(const char *msg)//DCL03-A
{
    char const *p=msg;  //DCL30-C
    //p is out of scope outside so it is compilant code
   //setColor(4);
    printf(p);
    slowprint(14,"\nWhat is this! The result is a bit unexpected!..\nwhat to do!!");
   Sleep(2000);
    slowprint(14,"\nTime to call the technicians!..");
    getchar();
    slowprint(6,"\nfixing..");
    return -3;
}

short getDigits(short num)
{
    short d=0;
    while(num)
    {
        num=num/10;
        d+=1;
    }
    return d;
}
```

launch.c

```
#include"headers.h"

player displayStartScreen1()
{
    //setColor(15);
    //setBackgroundColor(0);
    // saveDefaultColor();
    char badchars[]="@%^&*#$?()|";
    int count=0;
    player p;
    do
    {
        system("cls");
       //setColor(11);
        line_displayer(14);
        printf("\t\t\t\t\t");
       //setBackgroundColor(1);
        flashy_text(14,"ROCKET LAUNCH ASSIST");
       //setBackgroundColor(0);
        printf("\n");
        line_displayer(14);
       //setColor(11);
        slowprint(11,"\n\n\nEnter name:");
        // scanf("%^[\n]s",p.name);
        // fflush(stdin);
        scanf("%19s",p.name); //STR34-C
        count=strspn(p.name,badchars);  //STR02-C
        if(count>0)
        {
          //setColor(4);
            printf("\n[STR02-C FAILED - SANITIZATION]");
            slowprint(4,"\nSorry please dont include special characters!\n");
          Sleep(2000);
          //setColor(11);
        }
    } while (count!=0);
   Sleep(2000);
   //setColor(6);
    printf("\n[FOLLOWING STR34-
C - DONT COPY UNBOUNDED DATA TO FIXED LENGTH ARRAY]");
   Sleep(2000);
   //setColor(11);
    getchar();
    return p;
}

player displayStartScreen2(player pl)
{
    system("cls");
    slowprint(14,"You're on a mission to launch a satellite to");
    slowprint(6," 'Planet Omicron'");
    dots(6,3);
    slowprint(10,"\nThis mission is to find extraterrestial life supporting planets..");
   //setColor(11);
    printf("\nAre you ready?? - Y/y :");
    fflush(stdin);
    char c=getchar();
    if(c=='Y' || c=='y')
    {
        fflush(stdin);//FIO09-A
        slowprint(3,"\nGood..\nNow please enter your payload Name(max 19 chars): ");
        fgets(pl.p.satellite_name, 20, stdin);// FIO37-C  STR31-C STR03-A
        char *loc=strchr(pl.p.satellite_name,'\n'); //FIO36-C
        *loc='\0';
       //setColor(6);
        printf("\n[FOLLOWING STR31-C - STRING STORAGE SPACE]");
        printf("\n[FOLLOWING STR03-A - STRING TRUNCATION]");
       //setColor(3);
        printf("\nPayload Name: %s",pl.p.satellite_name);
       Sleep(2000);
        // printf("\nChoose Payload weight :\n1.upto 1500kg\n2.1500-2000kg\n3.2000-4000kg");
        // printf("Enter choice");
        // scanf("%d",&ch);
        printf("\n");
        flashy_text(10,"Great! Let's get started with your mission..");
        printf("\n");
        fflush(stdin);
        getchar();
        return pl;
    }
    exit(0);
}

player decideRocket(player pl)
{
    int ch;
    system("cls");
    slowprint(11,"\t1.PSLV - XL");
```

```
//setColor(14);
 printf("\nHeight:44m\nStart Thrust:300kN\nPropulsion:Solid +Liquid");
 printf("\nPayload limit:1500kg\nFuel Capacity: 3000 units");
 slowprint(11,"\n\n\t2.RLV-TD");
//setColor(14);
 printf("\nHeight:48m\nStart Thrust:350kN\nPropulsion:Liquid");
 printf("\nPayload limit:2000kg\nFuel Capacity: 3500 units");
 slowprint(11, "\n\n\t3.GSLV-Mark III");
//setColor(14);
 printf("\nHeight:43.43m\nStart Thrust:500kN\nPropulsion:Solid+Liquid+Cryogenic");
 printf("\nPayload limit:4000kg\nFuel Capacity: 5000 units");
//setColor(10);
 printf("\n\nEnter your choice: ");
 // resetColor();
 scanf("%d",&ch);
 // pl.p.weight = ch == 1 ? 1500 : ch == 2 ? 2000 : 4000;
 switch (ch)
 {
 case 1:
    pl.r.fuel_capacity=3000.0;
    pl.r.height=44.0;
    pl.r.start_thrust=300.0;
    strcpy(pl.r.propulsion,"Solid+Liquid");
    strcpy(pl.r.rocket_type,"PSLV-XL");
    pl.p.weight=1500;
    break;
 case 2:
    pl.r.fuel_capacity = 3500.000;
    pl.r.height = 48;
    pl.r.start_thrust = 350.0;
    strcpy(pl.r.propulsion, "Liquid");
    strcpy(pl.r.rocket_type, "RLV-TD");
```

```
        pl.p.weight = 2000;
        break;
    case 3:
        pl.r.fuel_capacity = 5000.00;
        pl.r.height = 43.43;
        pl.r.start_thrust = 500.0;
        strcpy(pl.r.propulsion, "Solid+Liquid+Cryogenic");
        strcpy(pl.r.rocket_type, "GSLV-Mark III");
        pl.p.weight = 1500;
        break;
    // default:
    //     break;
    }
   //setColor(7);
    printf("\nGood..Now lets start the actual Mission..");
    fflush(stdin);
    getchar();
    return pl;
}
```

payload.c

```
#include"headers.h"



//EXP00-A PRE00-A
#define capability_fail(t1,t2) t1/10*t1-t2/10*t2 ==0 ? 1 :0
```

```
static short altitude;


short countdown(player pl)
{
    system("cls");
    slowprint(11,"Assessing rocket capability..");
   Sleep(1500);
    int check=capability_fail(pl.r.fuel_capacity,pl.r.start_thrust);
   //setColor(12);
    printf("\n%s",(check==1 ? "Capable" : "Not Capable"));
    error_disp("\n[EXP00-A FAILED- PARENTHESES]\n[PRE00-A FAILED-
INLINE AGAINST MACROS]");
    check=capability_pass(pl);
   //setColor(10);
    printf("\n%s",(check==1 ? "Rocket now Capable" : "Not Capable"));
    printf("\nIssue fixed!");
   Sleep(1000);
    fflush(stdin);
    getchar();
   //setColor(11);
    printf("\nRocket countdown..begins..NOW!\n");
   Sleep(500);
    short c;
    while (c != -1)
    {
        printf("%d", c);
       Sleep(1000);
        printf("\b");
        c--;
    }
    error_disp("\n[EXP33-C FAILED - UNINITIALIZED VARIABLES]");
```

```
//setColor(10);
 printf("\nIssue fixed!\n");
 fflush(stdin);
 getchar();
 system("cls");
 flashy_text(11,"Rocket countdown begins NOW!!");
//setColor(14);
 c=9;
 printf("\n");
 while(c!=-1)
 {
    printf("%d",c);
   Sleep(1000);
    printf("\b");
    c--;
 }
Sleep(1000);
 slowprint(10,"\nRocket ignition successful!");
Sleep(1000);
 slowprint(14,"\nRocket acheving escape velocity..");
 float vel=getVelocity(pl);
 printf("\nRocket's velocity: %lf",vel);
 if(!(vel==11))
 {
   //setColor(12);
    printf("\nSeems like the rocket is failing..");
    printf("\n[FLP32C - DOMAIN ERRORS IN MATH FUNCTIONS FAILED]!");
   //setColor(10);
    printf("\nFixing..\n");
    flashy_text(10,"Sucessful!");
    getchar();
 }
```

```
    system("cls");
    slowprint(11,"\nRocket velocity now =11km/s.. Escape velocity reached!");
   Sleep(1000);
    slowprint(11,"\nRocket is in motion NOW!..SUCCESS!");
    //sleep(1500);
    fflush(stdin);
    getchar();
    short success=1;
    return success;
}

short phase1(player pl)
{
    system("cls");
    slowprint(3,"Entering rocket deploy monitor page...");
   Sleep(2000);
    system("cls");
    line_displayer(14);
   //setColor(11);
    printf("\t\t\t\t\t\tROCKET MONITOR CENTRE\n");
    line_displayer(14);
   Sleep(1500);
   //setColor(10);
    printf("\nRocket altitude :");
   //setColor(11);
    while(altitude!=300)
    {
        printf("%d",altitude);
        printf("hm");
        printf("\b\b");
        short d=getDigits(altitude);
        while(d!=0)
        {
            printf("\b");
            d--;
        }
        if(altitude==0)
            printf("\b");
        altitude+=50;
       Sleep(1000);
    }
   Sleep(1000);
    printf("\nRocket reached optimal altitude for phase1..");
   Sleep(1000);
    printf("\nLower body seperation initiated..");
   Sleep(1000);
    printf("\n");
    flashy_text(10,"SUCCESSFUL!!");
   Sleep(1000);
    slowprint(6,"\nEntering PHASE 2..");
    short ph=1;
   Sleep(1000);
    return ph;
}
short phase2(player pl)
{
    system("cls");
    line_displayer(14);
   //setColor(11);
    printf("\t\t\t\t\t\tROCKET MONITOR CENTRE\n");
    line_displayer(14);
   //setColor(10);
   Sleep(1500);
    printf("\nRocket altitude :");
```

```
  //setColor(11);
   while(altitude!=600)
   {
       printf("%d",altitude);
       printf("hm");
       printf("\b\b");
       short d=getDigits(altitude);
       while(d!=0)
       {
           printf("\b");
           d--;
       }
       if(altitude==0)
           printf("\b");
       altitude+=50;
      Sleep(1000);
   }
  Sleep(1000);
   printf("\nRocket reached optimal altitude for phase2..");
  Sleep(1000);
   printf("\nMiddle body seperation initiated..");
  Sleep(1000);
   printf("\n");
   flashy_text(10, "SUCCESSFUL!!");
  Sleep(1000);
   slowprint(6, "\nEntering PHASE 3..");
   short ph=1;
  Sleep(1000);
   return ph;
}
short phase3(player pl)
{
   system("cls");
   line_displayer(14);
  //setColor(11);
   printf("\t\t\t\t\t\tROCKET MONITOR CENTRE\n");
   line_displayer(14);
  //setColor(10);
  Sleep(1500);
   printf("\nRocket altitude :");
  //setColor(11);
   while(altitude!=700)
   {
       printf("%d",altitude);
       printf("hm");
       printf("\b\b");
       short d=getDigits(altitude);
       while(d!=0)
       {
           printf("\b");
           d--;
       }
       if(altitude==0)
           printf("\b");
       altitude+=10;
      Sleep(1000);
   }
  Sleep(1000);
  //setColor(11);
   printf("\nRocket reached deploy Point!!");
  Sleep(1000);
   slowprint(13,"\nInitiating satellite deploy..\n");
  Sleep(1000);
   flashy_text(10,"SUCCESSFUL!!");
```

```
  Sleep(1000);
   short ph=1;
  Sleep(1000);
   return ph;
}
```

find_life.c

```
#include"headers.h"

void launchtest_run()
{
   //STR00-A   STR01-A
   slowprint(9,"\nThis will be over soon..");
  Sleep(1000);
   string_m a=NULL;
   char *cstr;
   strcreate_m(&a,"Hi demo translation!",0,NULL);
   cstr=getstrptr_m(a);
   printf("\n");
  //setBackgroundColor(13);
   slowprint(11,cstr);
  //setBackgroundColor(0);
   free(cstr);
   slowprint(6,"\n[ABIDING STR00-
A USE TR24731 STANDARD FOR STRING MANIPULATION]");
   slowprint(6,"\n[ABIDING STR01-A USING MANANGED LIBRARY ]");
   getchar();
}
```

```
short payloadDeployed(player pl)
{
    system("cls");
    slowprint(10,"Good job!..Now we'll actually find if the rumours of life on Planet Omicron are
True..");
   Sleep(2000);
    slowprint(11,"\nNow this launch assisst will help you in communicating with your satellite de
ployed..");
   Sleep(2000);
    slowprint(11,"\nYour satellite will be sending messages at periodic intervals of time..");
   Sleep(2000);
    slowprint(6,"\nOver to satellite communication protocol please..");
    getchar();
    system("cls");
    line_displayer(10);
    printf("\t\t\t\t\t\t");
   //setColor(13);
   //setBackgroundColor(14);
    printf("SATELLITE COMMUNICATION PROTOCOL");
   //setBackgroundColor(0);
    printf("\n\n");
    line_displayer(10);
    slowprint(11,"\nEnabling AI assisted Translation & Decoding Protocol..");
   Sleep(1000);
    slowprint(13,"\nSuccesfully Enabled!");
   Sleep(1000);
    slowprint(6,"\nDoing a test Run");
    dots(6,2);
   Sleep(2000);
    launchtest_run();
    printf("\nTest Run SUCCESS!");
```

```
Sleep(1000);
 fflush(stdin);
 getchar();
 system("cls");
 line_displayer(10);
 printf("\t\t\t\t\t\t");
//setColor(13);
//setBackgroundColor(14);
 printf("SATELLITE COMMUNICATION PROTOCOL");
//setBackgroundColor(0);
 printf("\n\n");
 line_displayer(10);
Sleep(500);
//setColor(11);
 printf("\n%s",pl.p.satellite_name);
//setBackgroundColor(13);
 slowprint(11,": Arcane microbiological life detected..Sending genomes");
//setBackgroundColor(0);
Sleep(3000);
//setColor(11);
 printf("\n\n%s", pl.p.satellite_name);
//setBackgroundColor(13);
 slowprint(11, ": Analyzing genome samples obtained");
//setBackgroundColor(0);
Sleep(3000);
//setColor(11);
 printf("\n\n%s", pl.p.satellite_name);
//setBackgroundColor(13);
 slowprint(11, ": Genome (RNA-ATCU schema) sequence being sent");
//setBackgroundColor(0);
Sleep(3000);
//setColor(8);
```

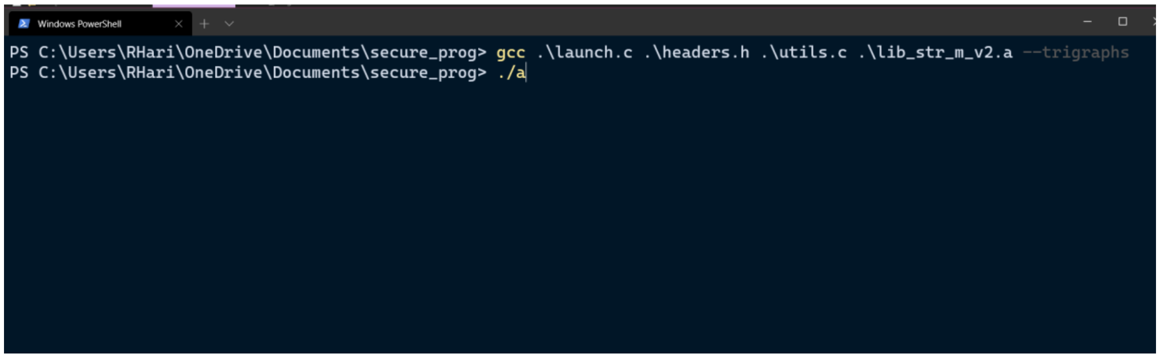
```
   printf("\nExiting satellite protocol..");
  Sleep(3000);
   slowprint(11,"\nHere's the Genome sequence obtained..");
   char genomesequence[] = "gcauacac??'??]??)??(ucgcuaugucgauaacaac";
   printf("\n");
   slowprint(13,genomesequence);
  Sleep(2000);
   slowprint(6,"\nWait..what!? Genome sequence is corrupted!\n");
  Sleep(1000);
  //setColor(12);
   printf("\n[PRE05-A FAILED TRIGRAPHS CHECK]");
  Sleep(2000);
  //setColor(11);
   printf("\nOkay time to call the technicians..");
  Sleep(2000);
   printf("\nFixing the problem..");
   char new_genomesequence[] = "gcauacac??ucgcuaugucgauaacaac";
   getchar();
   system("cls");
   slowprint(14,"\nNew Sequence: ");
  //setColor(13);
   printf("%s",new_genomesequence);
  Sleep(5000);
   getchar();
   char buffer[80];
   sprintf(buffer,"\nOkay Great Work Scientist %s!\nYou've found extraterrestrial life!\n", pl.na
me);
   slowprint(11, buffer);
  Sleep(3000);
   char buff[30];
   sprintf(buff, "*Praises and shouts for Researcher %s! *", pl.name);
   printf("\n");
```

```
    flashy_text(11,buff);
   Sleep(2000);
    getchar();
    slowprint(14,"\nOne Thing.. Please forgive the programmers for desigining this horrible Rocket Launch Assist!");
    getchar();
    printf("\nGoodbye %s. Your mission with payload %s is SUCCESSFUL..",pl.name,pl.p.satellite_name);
    printf("\nUntil next Time..");
    getchar();
    return 1;
}
```

5.0 TESTING

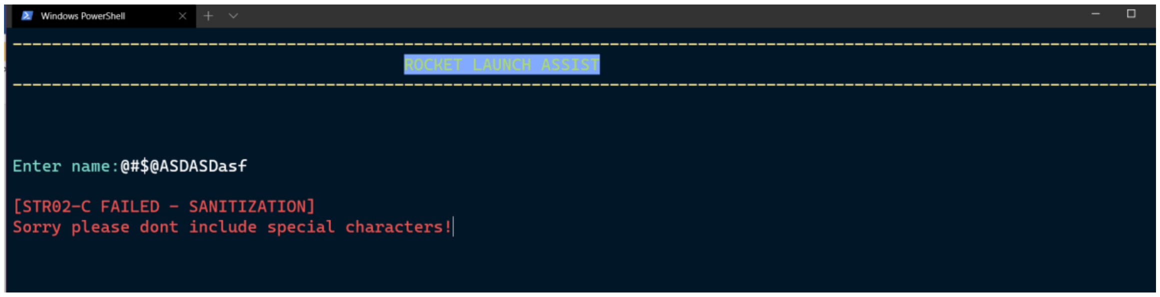
5.1 LAUNCH PHASE

Compiling software



```
PS C:\Users\RHari\OneDrive\Documents\secure_prog> gcc .\launch.c .\headers.h .\utils.c .\lib_str_m_v2.a --trigraphs
PS C:\Users\RHari\OneDrive\Documents\secure_prog> ./a
```

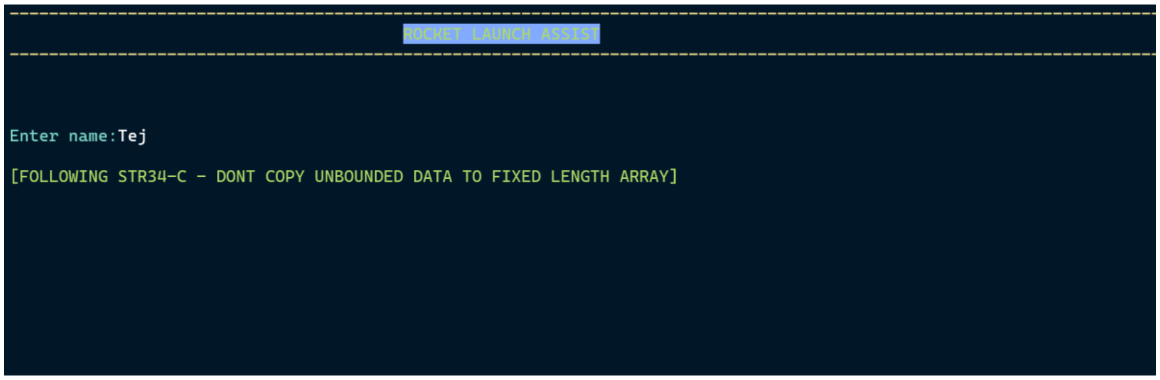
The input data which the name of the controller is checked for Data Sanitization and to avoid copying string to array of fixed length.



```
------------------------------------------------------------------------------------------------------------------------
                                               ROCKET LAUNCH ASSIST
------------------------------------------------------------------------------------------------------------------------


Enter name:@#$@ASDASDasf

[STR02-C FAILED - SANITIZATION]
Sorry please dont include special characters!
```

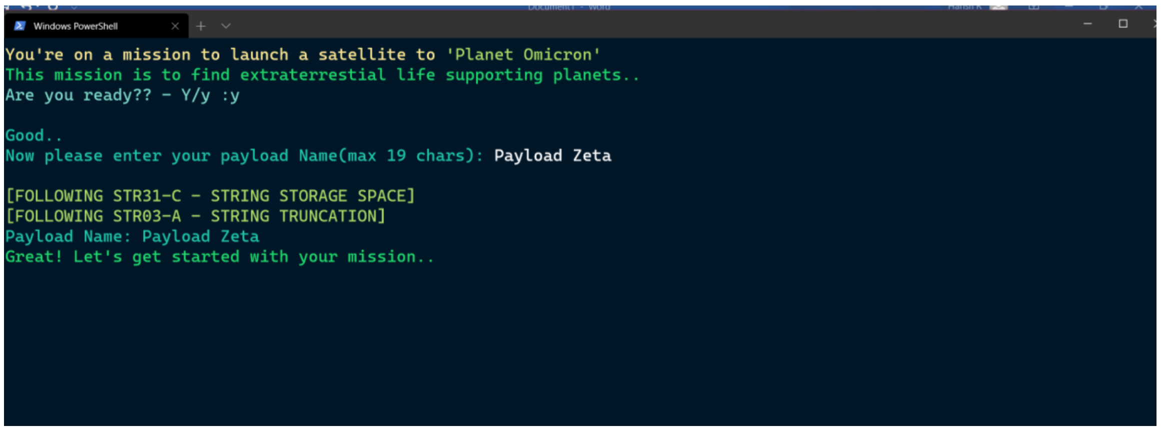


```
------------------------------------------------------------------------------------------------------------------------
                                               ROCKET LAUNCH ASSIST
------------------------------------------------------------------------------------------------------------------------


Enter name:Tej

[FOLLOWING STR34-C - DONT COPY UNBOUNDED DATA TO FIXED LENGTH ARRAY]
```

While input is taken for payload name we warn the user to input data less than maxlength defined, else extra data will be truncated and the controller of the project will not be able to use the payload’s name to access payload during the mission.



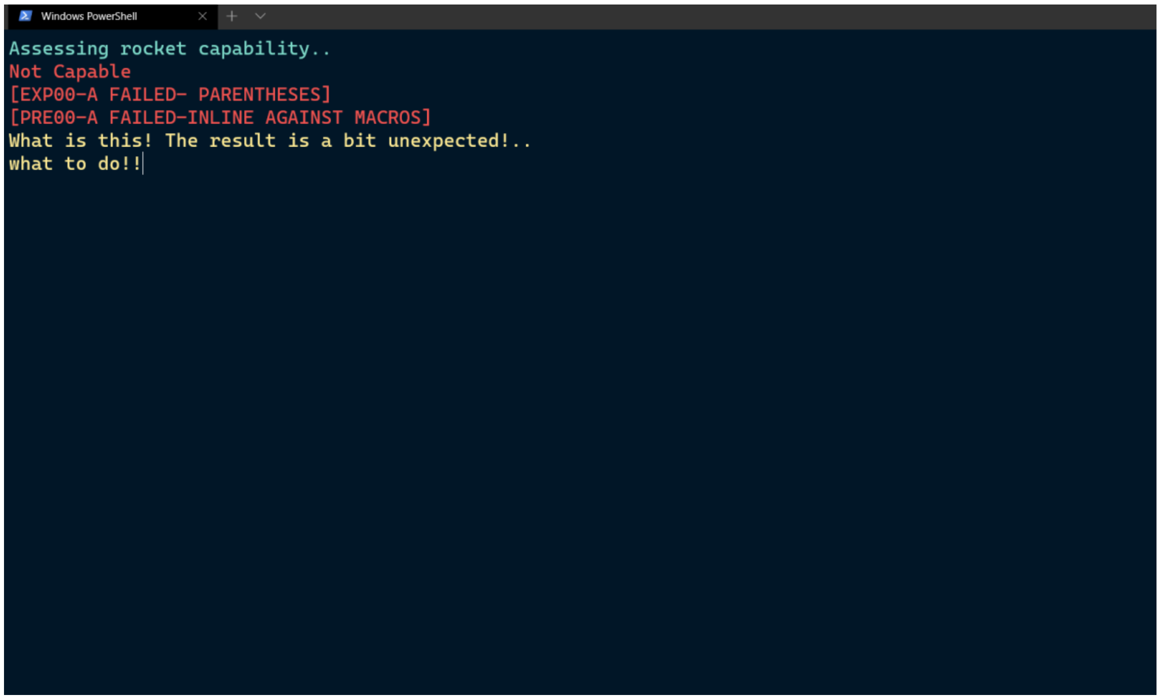
```
You're on a mission to launch a satellite to 'Planet Omicron'
This mission is to find extraterrestial life supporting planets..
Are you ready?? - Y/y :y

Good..
Now please enter your payload Name(max 19 chars): Payload Zeta

[FOLLOWING STR31-C - STRING STORAGE SPACE]
[FOLLOWING STR03-A - STRING TRUNCATION]
Payload Name: Payload Zeta
Great! Let's get started with your mission..
```

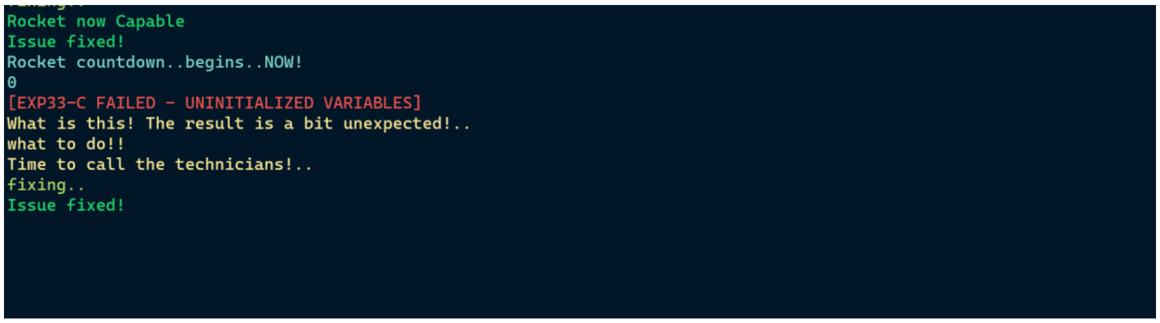
5.2 PARTITION PHASE

During the capability check, the resulting calculation leads a wrong answer because of non paranthesizing the expression while evaluation and also non-priotizing due to macro-induced errors.



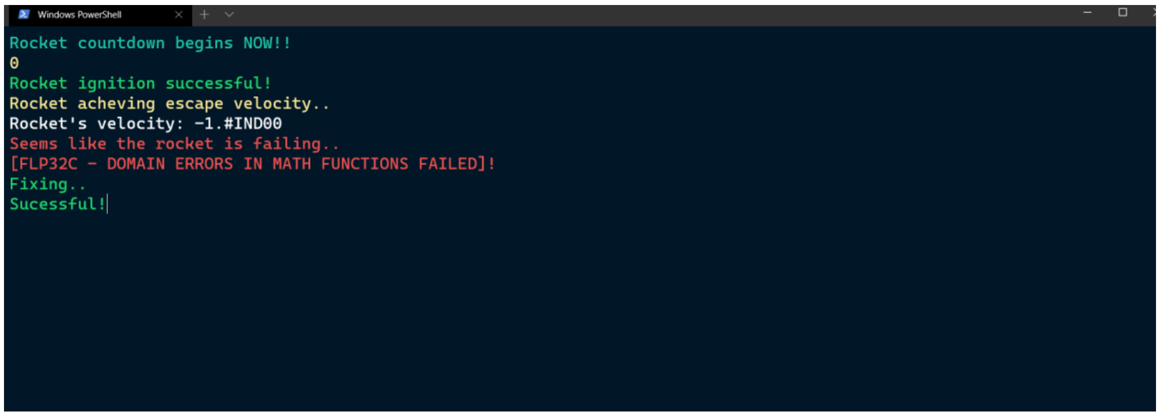
```
Assessing rocket capability..
Not Capable
[EXP00-A FAILED- PARENTHESES]
[PRE00-A FAILED-INLINE AGAINST MACROS]
What is this! The result is a bit unexpected!..
what to do!!
```

During countdown of the rocket countdown remains zero from start. Thus it is known that the variables have not been initialized. We fix this by initializing the variable with certain countdown value.



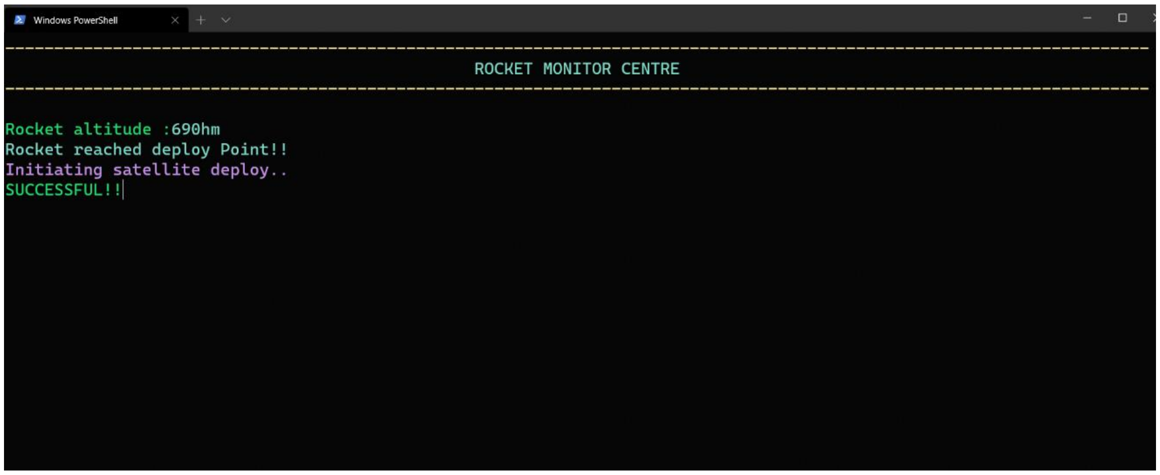
```
Rocket now Capable
Issue fixed!
Rocket countdown..begins..NOW!
0
[EXP33-C FAILED - UNINITIALIZED VARIABLES]
What is this! The result is a bit unexpected!..
what to do!!
Time to call the technicians!..
fixing..
Issue fixed!
```

After the rocket is launched, we see there is a negative velocity appearing on the screen. This was due to domain errors in the calculation. Due to a negative value in the square root function a domain error was raised. This has been solved by appropriate calculation of the escape velocity



```
Rocket countdown begins NOW!!
0
Rocket ignition successful!
Rocket acheving escape velocity..
Rocket's velocity: -1.#IND00
Seems like the rocket is failing..
[FLP32C - DOMAIN ERRORS IN MATH FUNCTIONS FAILED]!
Fixing..
Sucessful!
```

Lower part, middle part then payload gets deployed from the body after reaching the correct altitude for payload deployment.

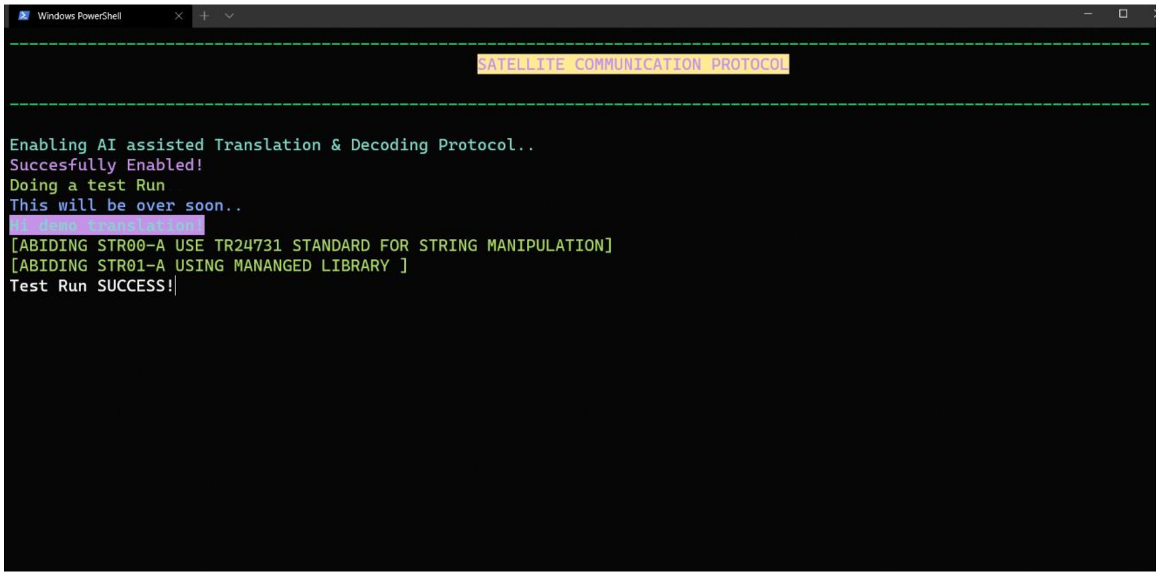


```
------------------------------------------------------------------------------------------------------------------------
                                              ROCKET MONITOR CENTRE
------------------------------------------------------------------------------------------------------------------------

Rocket altitude :690hm
Rocket reached deploy Point!!
Initiating satellite deploy..
SUCCESSFUL!!
```

5.3 SEARCH PHASE

We use managed string library to ensure proper data transmission by the payload deployed in Planet Omricon. To ensure that sensitive data is not corrupted managed strings library have been used for safety of the genome sequence.

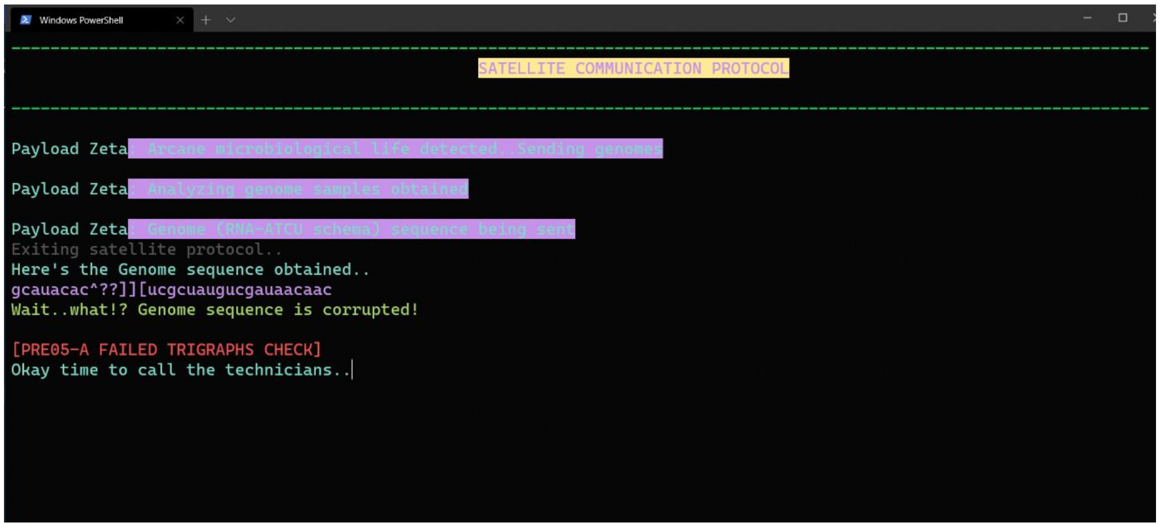


```
------------------------------------------------------------------------------------------------------------------------
                                              SATELLITE COMMUNICATION PROTOCOL

------------------------------------------------------------------------------------------------------------------------

Enabling AI assisted Translation & Decoding Protocol..
Succesfully Enabled!
Doing a test Run
This will be over soon..
Hi demo translation!
[ABIDING STR00-A USE TR24731 STANDARD FOR STRING MANIPULATION]
[ABIDING STR01-A USING MANANGED LIBRARY ]
Test Run SUCCESS!
```

We found that even though managed string functions are used in the data collection, some data is corrupted while transferring. We see a trigraph issue here and fix it to conform life on that planet.



```
--------------------------------------------------------------------------------------------------------------------
                                              SATELLITE COMMUNICATION PROTOCOL

--------------------------------------------------------------------------------------------------------------------

Payload Zeta: Arcane microbiological life detected..Sending genomes

Payload Zeta: Analyzing genome samples obtained

Payload Zeta: Genome (RNA-ATCU schema) sequence being sent
Exiting satellite protocol..
Here's the Genome sequence obtained..
gcauacac^??]][ucgcuaugucgauaacaac
Wait..what!? Genome sequence is corrupted!

[PRE05-A FAILED TRIGRAPHS CHECK]
Okay time to call the technicians..
```

6.0 SCALABILITY AND MAINTAINANCE

Many functions are ideal to be used directly like the functions for aesthetic display such as flashytext, dots etc.. they can be directly used in many modules without any changes at all.

Functions such as phases and payload deployment all are easily expandable for logic as they serve as ‘wrapper functions’ whose logic can be easily extended without conflicts because of the modular approach chosen where every possible step in rocket launch is broken down into its own functions.

Due to modularity debugging is very easy as the errors can be easily isolated and the fact that there is very little complex logic and pointer arithmetic and pointer manipulation makes this bulletproof to several segmentation errors and stack overflows. Strings are the difficult part to handle yet they have also been resolved by using managed string library functions which is a hard nut to crack . With such a ‘steel bulwark’ application for satellite monitoring an rocket configuration, scientists can now focus on the actual missions instead of looking where segmentation faults happened.

Modules enable future users to rewrite into their own payload details so there is no need to rewrite every time we deploy a new rocket. Changes can be made to existing functions by writing simple user defined functions, thus makes Scalable for any rocket launch.

Collecting genome sequence shows us that there might be corrupt data that will be received in different instances of different projects. This find_life.c module can be a model for trigraphs maintenance as well as maintaining code regarding transmission of details from the payload.

7.0 CONCLUSION

C stands as a developers' favorite in its speed and its handy low level application programming especially for applications such as the rocket launch endeavor. By using Secure Programming the showstopper issues of pointer manipulation buffer overflow string access and writes , file operations all are 'CERT-proofed' to guaranteed smooth and reliable operations in such costly missions where life and money is lost due to small errors that propagate due to bad programming.

Especially, in Rocket Launch Assist Corrupted data due to disturbances and memory leakages due to unexpected changes in the program and Memory allocations to pointers or structure stored variables are quiet common. We have handled some string vulnerabilities using managed string library.

Other vulnerabilites by arrays and memory have been appropriately handled and domain based floating point calculations whose integrity is ensured by following appropriate floating point and integer recommendations. This ensures that no C-based errors fail the rocket launch mission. Memory management has been ensured as well as other requirements such as modularity have been efficiently incorporated to ensure maximum reliability and scalability.

These security concerns are abated thanks to Secure Programming. Now we – Tejeshwar U and R Harish, with our mentor and pro-Green Programmer – Dr Kiran Sir have brought the inception of rocket launch and assist tools using C. So reader we encourage you to see the worth and strength of C and also bid adieu now as humble students encompassed with the knowledge of the ultimate tool -Secure C.

8.0 TRIVIA

- SpaceshipTwo VSS Enterprise crashed during a test flight and took lives due to a premature deployment of the feathering system(the reentry of the space shuttle into the atmosphere is managed by a shuttle that is shuttlecock feather shaped)
- The Russian Royuz is a multistage rocket(3 phases as our application details) and on the 3rd phase launches the Fregat stage where the payload is deployed . On May 22 2009 due to a small miscalculation the rocket lost altitude and the satellite was deployed incorrectly
- Falcon 1 – Space X's first rocket launch failed because an aluminium nut rusted before launch and spewed fuel on the rocket and the rocket's engine collapsed due to the fire after flying for 34 seconds
- Arianespace, a very successful rocket launch provider launched a pair of satellites on Jan 25 2018. Everything was going right until the signal was lost. Later the signal was found and luckily the satellites were in place. They were not in the right inclination and luckily had thrusters so they were veered into place. How did this happen? A wrong human input was supplied and access to the telemetry data and the monitoring protocol went off . No data validation was in place.

9.0 REFERENCES

1. CERT C Programming language Secure coding standard
2. Secure coding in C and C++ -Robert Seacord
3. Secure C programming slides and lecture notes by Dr Kiran
4. Wikipedia