Automatic Debugging of C Programs

(Improvising learning by modeling C compilation errors)

Synopsis:

GCC is one of the famous and widely used open source complier used to compile and debug C programs. Apparently, it has been noted and also highlighted (by GNU) the known problems that affect users of GCC. “Some of the problems may be due to bugs in other software, or some are missing features that are too much work to add, and some are places where people's opinions differ as to what is best”. In this project, we focus on the errors and warnings produce by GNU compiler, and analyze those in order to understand and develop a predictive model that helps us to pinpoint the root cause (actual cause, instead of a number of lines) of the respective error or warning. It is no secret that C happens to be one the most widely used languages for coding around the world. Despite the popularity, there does not exist a single predictive model which pinpoints the sources of the errors that we most commonly come across. This is the motivation behind the project. Once successfully completed, the model will be of great help in reducing the debugging efforts and costs and will help in a better understanding of the language and its usage.

The Project:

We started off by collecting various types of errors and warning produce by the GNU compiler, and then logging those errors and warnings in the database. In this process, we are trying to identify patterns between various errors and warnings in order to successfully give out remedies when we get to the predictive model.

Parallelly, we will apply Natural Language Processing (NLP) techniques in order to parse the errors and warnings produce by GCC; so that we can match the code-errors with our database and can predict flaw in the code accurately.

Our aim is to mainly focus on the errors and warnings produce by the GNU compiler (more than 400 buggy programs has been complied in order to understand the pattern in the error-warning logs), and next generating and implementing a predictive model which accurately map the error-warning and highlight the accurate cause of the bug. Once the database is complete, we will use student developed programs, developed as a part of the course, to train our predictive model and thereafter other programs for testing purpose, so that we ensure an efficient model to ensure accuracy. We have made sure to test as many different kinds of codes that we come across in C, so that we have a wide array of errors which are commonly encountered and we are more successful in finding out any patterns between different errors in C. We have also examined and analyzed various kinds of mutation testing tools so that we get varied errors from the same erroneous code.

Project Details:

Our project will try to give out the user the root source of the error or the very probable cause of it. We aim to achieve this by studying existing C errors that are achieved using random mutation and random student codes containing compilation errors of many kinds. The database generated in the process gives us the underlying patterns of C compilation errors which are generated due to one single mistake. Below are examples given illustrating such patterns. This predictive model will be trained by a random sample of 70-80% dataset and remaining will be to use as test data. This predictive model will be than deployed to run on working data and live codes where we will use it to give out summary of our original compilation errors. We

will be using NLP (preferably Stanford NLP) to parse the error strings generated as compilation error at this stage.

Finally, we aim to develop an automatic tool to visualize the errors and warning in C programs which will thereafter helpful to debug the programs and may improve the learning by providing a comprehensive compiling environment (e.g., students, practitioners, etc.) for the users. We also implement the “Options for Debugging Your Program or GCC” in our tool provided by GNU.

Success Criteria:

1. The tool recognizes the errors and shows an intended predictive output for the respective errors and warnings.
2. The training and testing of the earlier developed predictive model strengthen the database and result in incorporating more number of errors and warnings.
3. Tool help to debug your program as per the guidelines provided by GNU.

Road-map:

Till now, we have analyzed and collected the errors and warning logs of more than 400 programs, produced by GNU compiler. We have also identified several patterns in that and now implementing them in Java language. Meanwhile, we are keen to collect data for at least 1000 C programs.

The road-map for developing the project under GSoC is as follows.
Week 1: Understanding the organization work and Project Literature
Week 2: Understanding of the GNU compiler and compilation process
Week 3 and 4: Understanding of the type of errors and warning, and our developed predictive model
Week 5, 6, 7: Implementation of the predictive model; training and testing of the model
Week 8, 9: Implementation of the guidelines for debugging your program by GNU; writing test cases
Week 10, 11: Overall testing the tool under development
Week 12: Preparation of Installers, Preparation of Manual (doc/video)

Appendix:

Few Illustrations for the errors and warning logs are given below:

Program 1: Code

```
#include<stdio.h>
#include<string.h>
int main()
{int i;
char ar[100];
char ar2[100];
gets(ar);
int n=strlen(ar);
int j=0;
for(i=0;i<n;i++)
{
if(ar[i]=='a' || ar[i]== 'e' || ar[i]=='o' || ar[i]== 'i' || ar[i]=='u' || ar[i]==32)
continue;
else
```

```
{ar2[j]=ar[i];
j=j+1;
} }
int n2=strlen(ar2);
for(i=0;i<n2;i++)
{
printf("%c",ar2[i];}
for(j=0;j<(n2-1);j++)
{
for(i=0;i<(n2-1);i++)
{char fal=ar2[i];
if(ar2[i]>=ar2[i+1])
{ar2[i]=ar2[i+1];
ar2[i+1]=fal;
}}}
for(i=0;i<n2;i++)
{
printf("%c",ar2[i]);
}
return 0;
}
```

Error strings-

"prog.c:23:19: error: expected ')' before ';' token printf(""%c"",ar2[i];}"
"prog.c:23:20: error: expected ';' before '}' token printf(""%c"",ar2[i];}"

Cause: This compilation error pops up because of a missing close bracket in line 23.

Program 2: Code

```
#include<stdio.h>
// struct team
//{
//str pname[];
//str tname[];
//int avg;
//}
int main()
{
struct team
{
str pname[];
str tname[];
int avg;
}
struct team p1,p2,p3,p4,p5;
scanf("%s%s%d%s%s%d%s%s%d%s%s%d%s%s%d",&.p1,&.p1,&.p1,&.p2,&.p2,&.p2,&.p3
,&.p3,&.p3,&.p4,&.p4,&.p4,&p.p5,&.p5,&.p5);
printf("%s %s %d\n",pname.p1,tname.p1,avg.p1)
printf("%s %s %d\n",pname.p2,tname.p2,avg.p2);
printf("%s %s %d\n",pname.p3,tname.p3,avg.p3);
printf("%s %s %d\n",pname.p4,tname.p4,avg.p4);
printf("%s %s %d\n",pname.p5,tname.p5,avg.p5);
return 0 ;
```

```
}
```

Error strings:

```
prog.c:14:1: error: unknown type name 'str'
prog.c:15:1: error: unknown type name 'str'
prog.c:20:101: error: 'p' undeclared (first use in this function)
prog.c:21:21: error: 'pname' undeclared (first use in this function)
prog.c:21:30: error: 'tname' undeclared (first use in this function)
```

Cause: These errors are arising because of the incorrect variable type while declaring which causes other errors as undeclared variables.

Program 3: Code

```
nclude<stdio.h>
#include<string.h>
struct cricket
{
char name[100];
char team[10];
int avg;
}p[5],temp;
int main()
{
int i,j;
for(i=0;i<5;i++)
scanf("%s %s %d",p[i].name,p[i].team,&p[i].avg);
for (i = 1; i <5; i++)
{ for (j = 0; j < 5 - i; j++)
{
if (strcmp(p[j].team, p[j + 1].team) > 0)
{
temp = p[j];
p[j] = p[j + 1];
p[j + 1] = temp;
}
}
}
for(i=0;i<5;i++)
printf("%s %s %d\n",p[i].name,p[i].team,p[i].avg);
return 0;
}
```

Error Strings:

```
prog.c:1:7: error: expected '=', ',', ';', 'asm' or '__attribute__' before '<' token
/usr/include/string.h:47:8: error: unknown type name 'size_t'
/usr/include/string.h:50:56: error: unknown type name 'size_t'
/usr/include/string.h:59:18: error: unknown type name 'size_t'
/usr/include/string.h:66:42: error: unknown type name 'size_t'
/usr/include/string.h:69:56: error: unknown type name 'size_t'
/usr/include/string.h:96:48: error: unknown type name 'size_t'
/usr/include/string.h:133:39: error: unknown type name 'size_t'
/usr/include/string.h:141:9: error: unknown type name 'size_t'
/usr/include/i386-linux-gnu/bits/string2.h:945:17: error: unknown type name 'size_t'
```

/usr/include/i386-linux-gnu/bits/string2.h:946:17: error: unknown type name 'size_t'
/usr/include/i386-linux-gnu/bits/string2.h:949:3: error: unknown type name 'size_t'
/usr/include/i386-linux-gnu/bits/string2.h:1297:47: error: unknown type name 'size_t'
prog.c:17:8: error: unknown type name 'size_t'
prog.c:17:8: error: 'size_t' undeclared (first use in this function)
prog.c:17:8: error: expected expression before 'const'
prog.c:17:8: error: expected expression before 'const'
Cause: This set of error arises due to the missing stdio.h header in line 1.

Program 4: Code –

```
#include<stdio.h>
int main()
{
int a,i,j,k,c,q,m,n,x;
scanf("%d",&a);
c=a;
for(i=0;a!=0;i++)
{
a=a/10;
}
int z[i];
for(j=0;j<i;j++)
{
z[j]=c%10;
c=c/10;
}
for(m=0;m<i-1;m++)
{
for(n=0;n<i;n++)
{
if(z[n]>z[n+1]
{z[n+1]=q}}}
for(x=i-1;x>=0;x--)
printf("%d"
```

Error string:
prog.c:22:1: error: expected ')' before '{' token
prog.c:22:11: error: expected expression before '}' token
prog.c:24:1: error: expected ')' at end of input
Cause: This set of errors is generated due to mismatch of brackets. (Brackets missing at end).

Program 5: Code-

```
#include<stdio.h>
int main()
{
int a,b,c,x,n1;
scanf("%d",&a);
char s[30],temp;
c=strlen(s);
char s2[30];
s=getchar();
for(x=0;x<c,x++)
```

```
{
temp=s[x];
s2[x]='temp';
s2[x]=n1[x]+a;
}
putchar(s2);
return 0;
}
```

Error string:

prog.c:9:2: error: assignment to expression with array type
prog.c:10:16: error: expected ';' before ')' token
prog.c:14:10: error: subscripted value is neither array nor pointer nor vector

Cause: Incorrect assignment at line 9.

Integer data type treated as vector/ array/ pointers.