190905514 **MOHAMMAD TOFIK** **ROLL NO : 62**
BATCH-C3

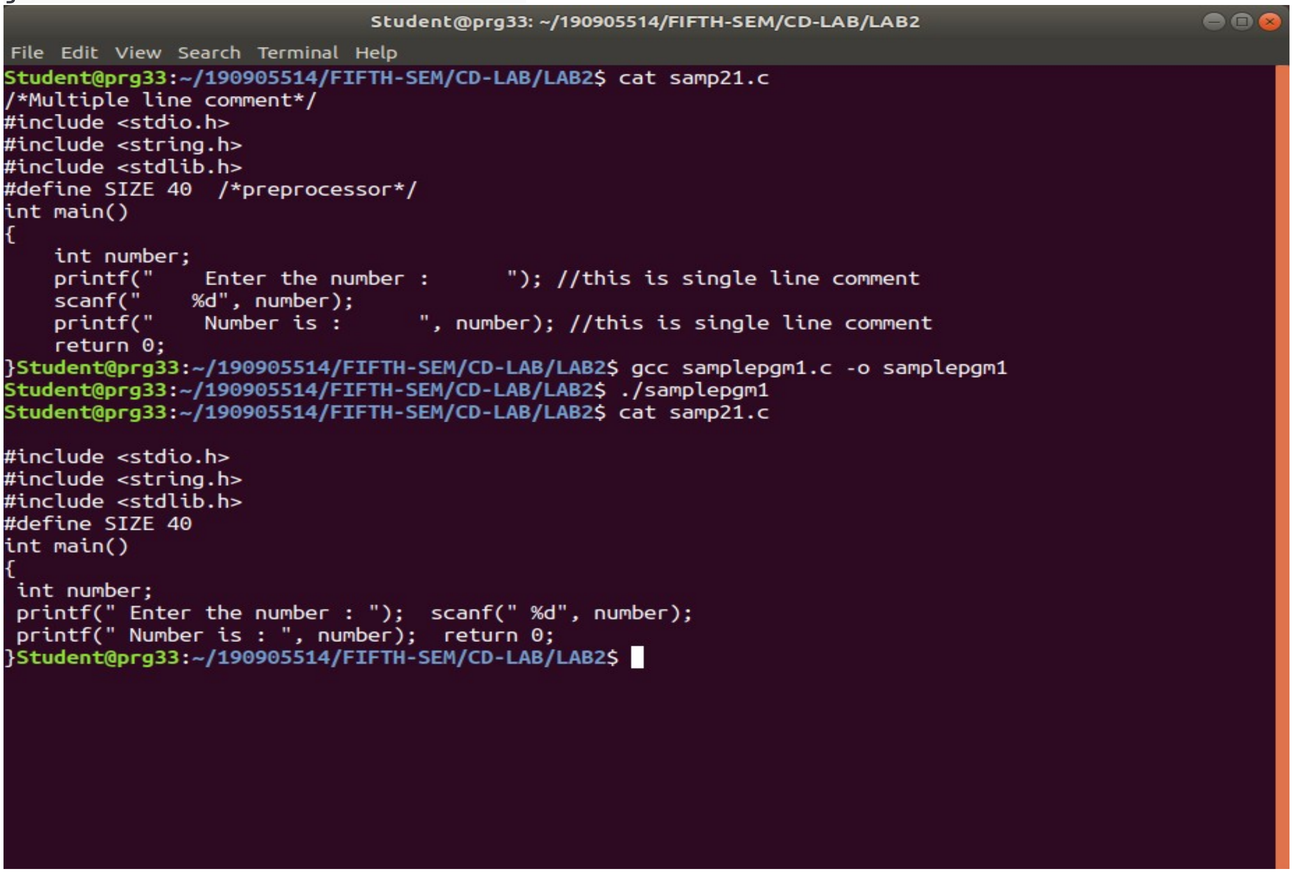
LAB – 2 :

sampleprogram :

1.Program to remove single and multiline comments from a given 'C' file.

```
#include <stdio.h>
#include<stdlib.h>
#include<string.h>
int main()
{
FILE *fa, *fb;
int ca, cb;
fa = fopen("prog.c", "r");
if (fa == NULL)
{
printf("Cannot open file \n");
exit(0);
}
fb = fopen("prog1.c", "w");
ca = getc(fa);
while (ca != EOF)
{
if (ca == ' ')
{
putc(ca, fb);
while (ca == ' ')
ca = getc(fa);
}
if (ca == '/')
{
cb = getc(fa);
if (cb == '/')
{
while (ca != '\n')
ca = getc(fa);
}
else if (cb == '*')
{
do
{
while (ca != '*')
ca = getc(fa);
ca = getc(fa);
} while (ca != '/');
}
else
{
putc(ca, fb);
putc(cb, fb);
}
}
else
putc(ca, fb);
ca = getc(fa);
}
fclose(fa);
fclose(fb);
return 0;
}
```

OUTPUT :



```
Student@prg33:~/190905514/FIFTH-SEM/CD-LAB/LAB2$ cat samp21.c
/*Multiple line comment*/
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#define SIZE 40   /*preprocessor*/
int main()
{
    int number;
    printf("    Enter the number :      "); //this is single line comment
    scanf("    %d", number);
    printf("    Number is :      ", number); //this is single line comment
    return 0;
}Student@prg33:~/190905514/FIFTH-SEM/CD-LAB/LAB2$ gcc samplepgm1.c -o samplepgm1
Student@prg33:~/190905514/FIFTH-SEM/CD-LAB/LAB2$ ./samplepgm1
Student@prg33:~/190905514/FIFTH-SEM/CD-LAB/LAB2$ cat samp21.c

#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#define SIZE 40
int main()
{
 int number;
 printf(" Enter the number : ");  scanf(" %d", number);
 printf(" Number is : ", number);  return 0;
}Student@prg33:~/190905514/FIFTH-SEM/CD-LAB/LAB2$ 
```

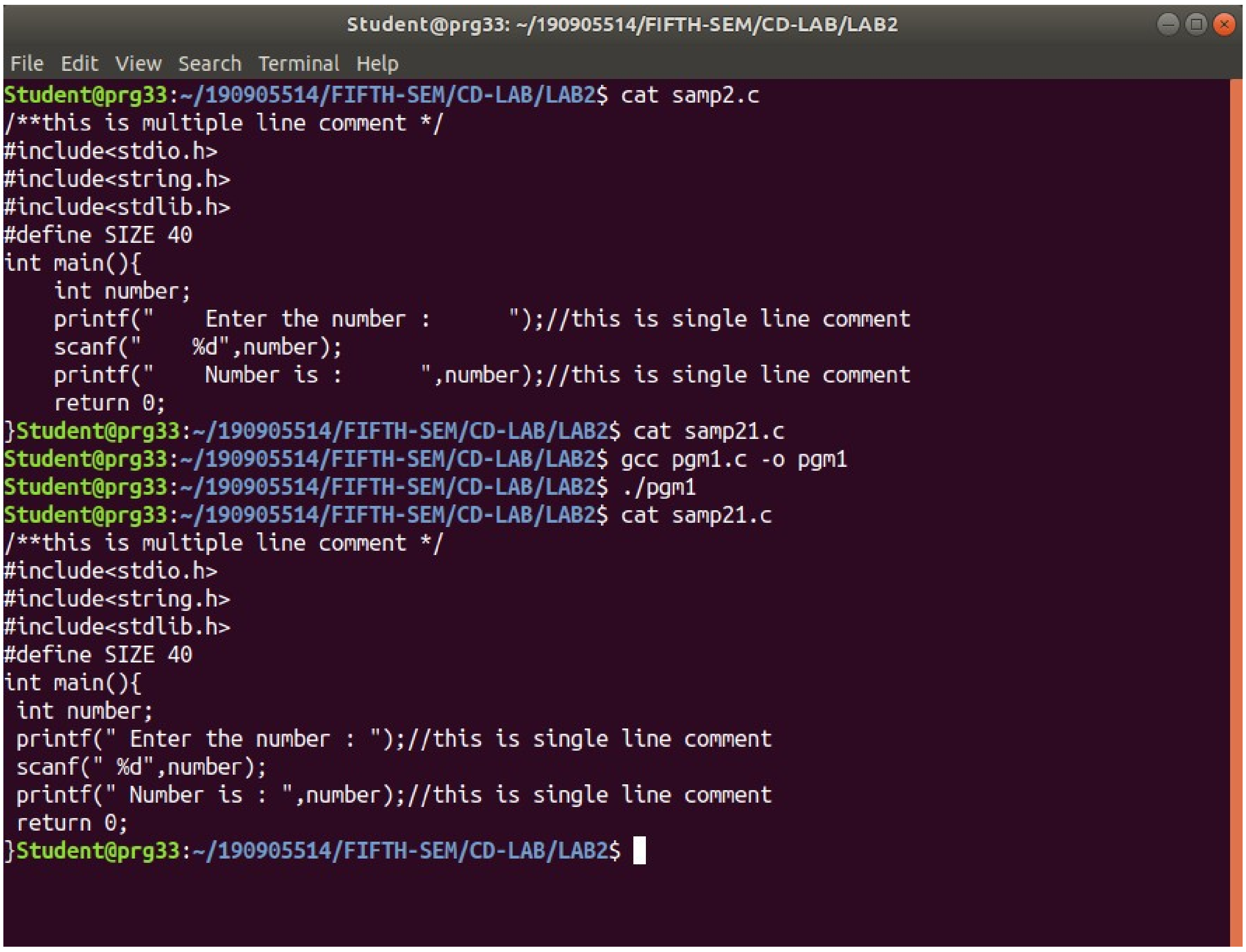
EXCERCISE :

1.That takes a file as input and replacesblank spacesand tabs by single space andwrites the output toa file.

pgm1.c

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
int flag = 0;
char ch;
FILE *fp1;
FILE *fp2;
fp1 = fopen("samp2.c", "r");
fp2 = fopen("samp21.c", "w");
if (fp1 == NULL || fp2 == NULL)
{
perror("File does not exist\n");
return 1;
}
while (1)
{
ch = fgetc(fp1);
if (ch == EOF)
{
break;
}
else if (!flag && (ch == ' ' || ch == '\t'))
{
fputc(' ', fp2);
flag = 1;
}
else if (!(ch == ' ' || ch == '\t'))
{
flag = 0;
fputc(ch, fp2);
}
}
fclose(fp1);
fclose(fp2);
}
```

OUTPUT :



```
Student@prg33:~/190905514/FIFTH-SEM/CD-LAB/LAB2$ cat samp2.c
/**this is multiple line comment */
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#define SIZE 40
int main(){
    int number;
    printf("    Enter the number :      ");//this is single line comment
    scanf("    %d",number);
    printf("    Number is :      ",number);//this is single line comment
    return 0;
}Student@prg33:~/190905514/FIFTH-SEM/CD-LAB/LAB2$ cat samp21.c
Student@prg33:~/190905514/FIFTH-SEM/CD-LAB/LAB2$ gcc pgm1.c -o pgm1
Student@prg33:~/190905514/FIFTH-SEM/CD-LAB/LAB2$ ./pgm1
Student@prg33:~/190905514/FIFTH-SEM/CD-LAB/LAB2$ cat samp21.c
/**this is multiple line comment */
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#define SIZE 40
int main(){
 int number;
 printf(" Enter the number : ");//this is single line comment
 scanf(" %d",number);
 printf(" Number is : ",number);//this is single line comment
 return 0;
}Student@prg33:~/190905514/FIFTH-SEM/CD-LAB/LAB2$
```

2.Todiscard preprocessor directives from the given input ‘C’ file.

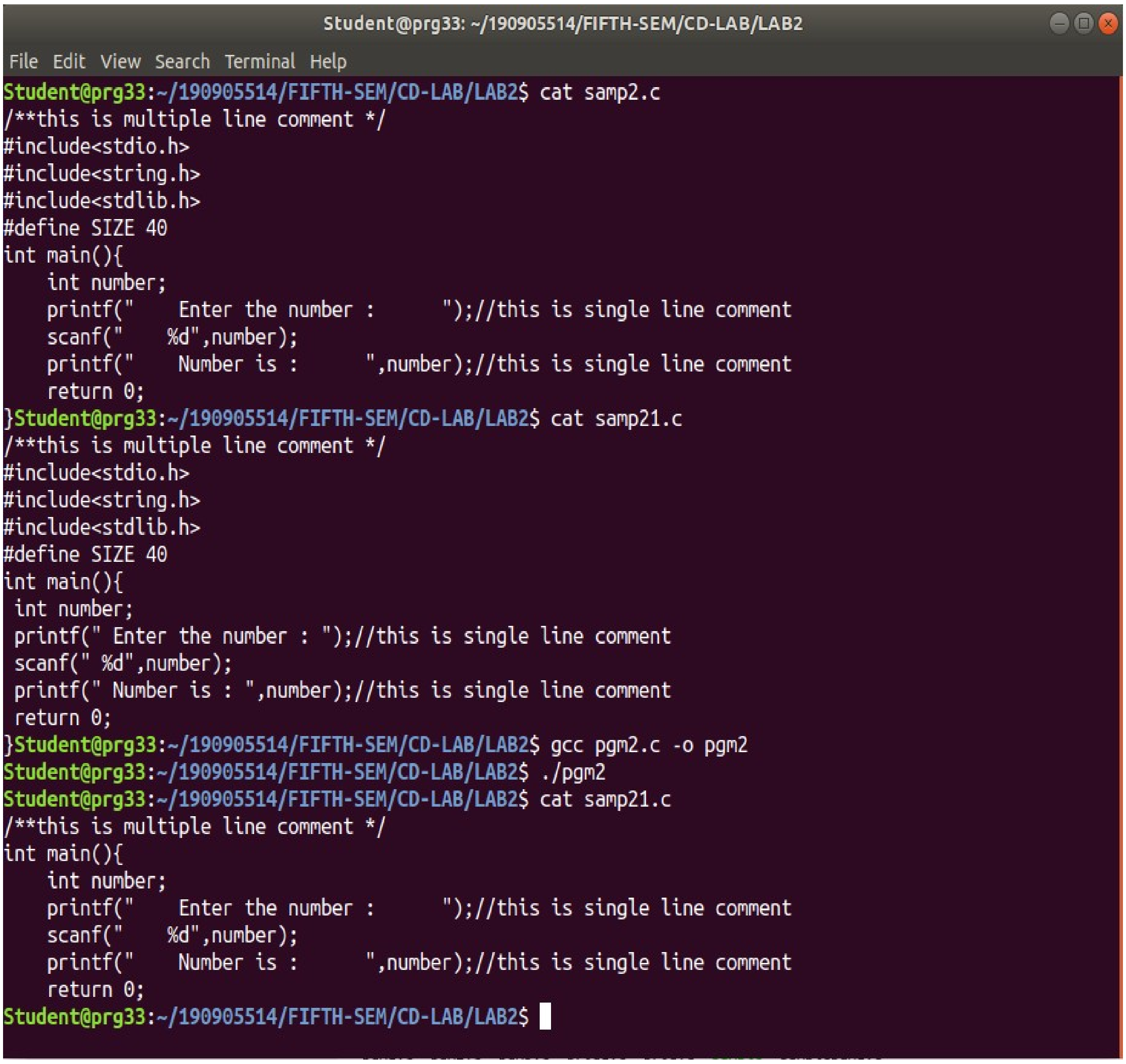
pgm2.c

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

#define FILEINPUT "samp2.c"
#define FILEOUTPUT "samp21.c"

const char *array[] = {"#include", "#define", "#if"};

int isDirective(const char *string)
{
```

```c
for (int i = 0; i < sizeof(array) / sizeof(char *); i++)
{
int len = strlen(array[i]);

if (strncmp(string, array[i], len) == 0)
{
return 1;
}
}

return 0;
}

int main()
{
char buffer[2048];
FILE *fp1;
FILE *fp2;
char ch;

fp1 = fopen(FILEINPUT, "r");
fp2 = fopen(FILEOUTPUT, "w");
if (fp1 == NULL || fp2 == NULL)
{
perror("File does not exist\n");
return 1;
}
while (fgets(buffer, 2048, fp1) != NULL)
{
if (!isDirective(buffer))
{
fputs(buffer, fp2);
}
}
fclose(fp1);
fclose(fp2);
fp1 = fopen(FILEINPUT, "w");
fp2 = fopen(FILEOUTPUT, "r");
ch = getc(fp2);
while (ch != EOF)
```

```
{
putc(ch, fp1);
ch = getc(fp2);
}
fclose(fp1);
fclose(fp2);
return 0;
}
```

OUTPUT :



```
Student@prg33:~/190905514/FIFTH-SEM/CD-LAB/LAB2$ cat samp2.c
/**this is multiple line comment */
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#define SIZE 40
int main(){
    int number;
    printf("    Enter the number :      ");//this is single line comment
    scanf("    %d",number);
    printf("    Number is :      ",number);//this is single line comment
    return 0;
}Student@prg33:~/190905514/FIFTH-SEM/CD-LAB/LAB2$ cat samp21.c
/**this is multiple line comment */
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#define SIZE 40
int main(){
 int number;
 printf(" Enter the number : ");//this is single line comment
 scanf(" %d",number);
 printf(" Number is : ",number);//this is single line comment
 return 0;
}Student@prg33:~/190905514/FIFTH-SEM/CD-LAB/LAB2$ gcc pgm2.c -o pgm2
Student@prg33:~/190905514/FIFTH-SEM/CD-LAB/LAB2$ ./pgm2
Student@prg33:~/190905514/FIFTH-SEM/CD-LAB/LAB2$ cat samp21.c
/**this is multiple line comment */
int main(){
    int number;
    printf("    Enter the number :      ");//this is single line comment
    scanf("    %d",number);
    printf("    Number is :      ",number);//this is single line comment
    return 0;
Student@prg33:~/190905514/FIFTH-SEM/CD-LAB/LAB2$ 
```

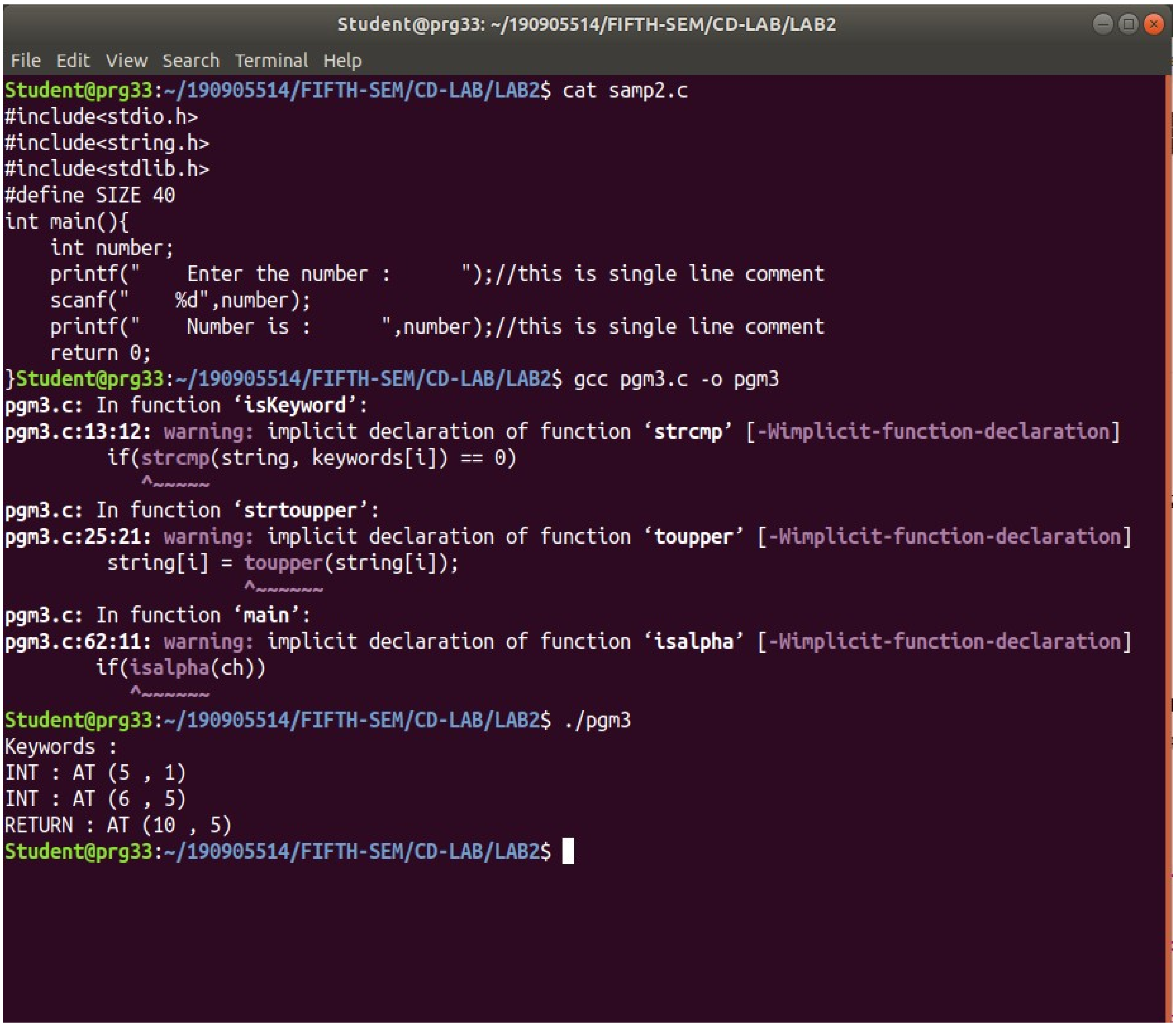
3.That takes C program as input, recognizesall the keywords and printsthem in upper case.

pgm3.c

```
#include <stdio.h>
#include <stdlib.h>
#define FILEINPUT "samp2.c"
const char *keywords[] = {"const", "char", "int","return", "for",
"while", "do", "switch",
"if", "else","case", "break"};
int isKeyword(const char *string)
{
for(int i = 0; i < sizeof(keywords)/sizeof(char *); ++i)
{
if(strcmp(string, keywords[i]) == 0)
{
return 1;
}
}
return 0;
}
void strtoupper(char *string, const int len)
{
for(int i = 0; i < len; ++i)
{
string[i] = toupper(string[i]);
}
}
enum
{
INSIDE_WORD,
OUTSIDE_WORD
};
int main()
{
FILE *fp1;
FILE *fp2;
int line=1;
int column=1;
int k=0;
```

```
char ch;
char buffer[512];
fp1 = fopen(FILEINPUT, "r");
if(fp1 == NULL)
{
perror("File does not exist\n");
return 1;
}
int state = OUTSIDE_WORD;
printf("Keywords : \n");
while((ch = fgetc(fp1)) != EOF)
{
switch(state)
{
case INSIDE_WORD:
if(isalpha(ch))
{
buffer[k++]=ch;
}
else
{
buffer[k]='\0';
if(isKeyword(buffer))
{
strtoupper(buffer, k);
printf("%s : AT (%d , %d)\n", buffer, line, column - k);
}
k=0;
state=OUTSIDE_WORD;
}
break;
case OUTSIDE_WORD:
if(isalpha(ch))
{
buffer[k++]=ch;
state=INSIDE_WORD;
}
break;
}
if(ch == '\n')
{
++line;
column = 1;
}
else
{
++column;
}
}
fclose(fp1);
}
```

OUTPUT :



```
Student@prg33:~/190905514/FIFTH-SEM/CD-LAB/LAB2$ cat samp2.c
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#define SIZE 40
int main(){
    int number;
    printf("    Enter the number :      ");//this is single line comment
    scanf("    %d",number);
    printf("    Number is :      ",number);//this is single line comment
    return 0;
}Student@prg33:~/190905514/FIFTH-SEM/CD-LAB/LAB2$ gcc pgm3.c -o pgm3
pgm3.c: In function ‘isKeyword’:
pgm3.c:13:12: warning: implicit declaration of function ‘strcmp’ [-Wimplicit-function-declaration]
         if(strcmp(string, keywords[i]) == 0)
            ^~~~~~
pgm3.c: In function ‘strtoupper’:
pgm3.c:25:21: warning: implicit declaration of function ‘toupper’ [-Wimplicit-function-declaration]
         string[i] = toupper(string[i]);
                     ^~~~~~~
pgm3.c: In function ‘main’:
pgm3.c:62:11: warning: implicit declaration of function ‘isalpha’ [-Wimplicit-function-declaration]
        if(isalpha(ch))
           ^~~~~~~
Student@prg33:~/190905514/FIFTH-SEM/CD-LAB/LAB2$ ./pgm3
Keywords :
INT : AT (5 , 1)
INT : AT (6 , 5)
RETURN : AT (10 , 5)
Student@prg33:~/190905514/FIFTH-SEM/CD-LAB/LAB2$
```

LAB-1 : QUESTION NUMBER 4 :

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

#define CAPACITY 50000 // Size of the Hash Table

unsigned long hash_function(char *str)
{
unsigned long i = 0;
for (int j = 0; str[j]; j++)
i += str[j];
return i % CAPACITY;
}

typedef struct Ht_item Ht_item;

// Define the Hash Table Item here
struct Ht_item
{
char *key;
char *value;
};

typedef struct HashTable HashTable;

// Define the Hash Table here
struct HashTable
{
// Contains an array of pointers
// to items
Ht_item **items;
int size;
int count;
};

Ht_item *create_item(char *key, char *value)
```

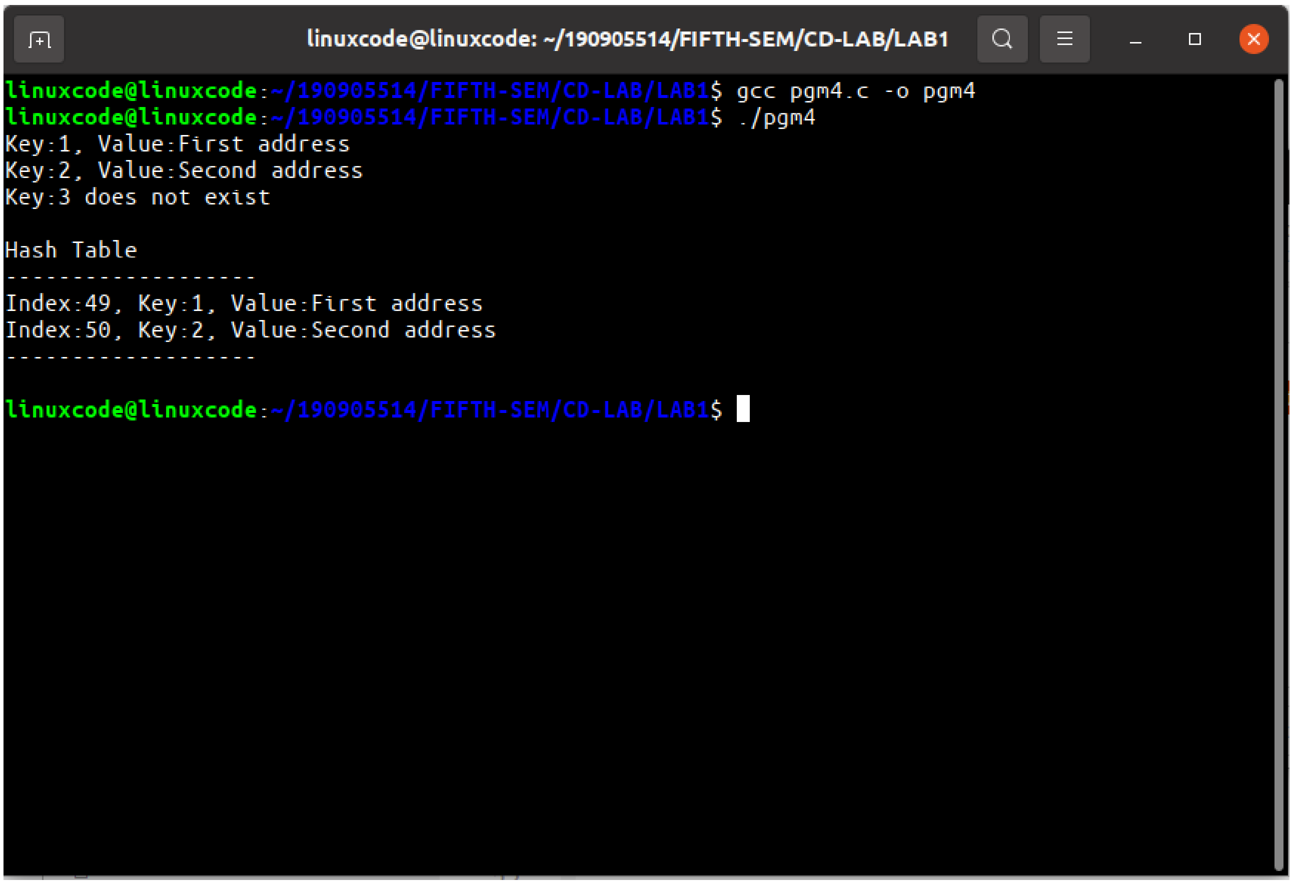
```c
{
// Creates a pointer to a new hash table item
Ht_item *item = (Ht_item *)malloc(sizeof(Ht_item));
item->key = (char *)malloc(strlen(key) + 1);
item->value = (char *)malloc(strlen(value) + 1);

strcpy(item->key, key);
strcpy(item->value, value);

return item;
}

HashTable *create_table(int size)
{
// Creates a new HashTable
HashTable *table = (HashTable *)malloc(sizeof(HashTable));
table->size = size;
table->count = 0;
table->items = (Ht_item **)calloc(table->size, sizeof(Ht_item *));
for (int i = 0; i < table->size; i++)
table->items[i] = NULL;

return table;
}

void free_item(Ht_item *item)
{
// Frees an item
free(item->key);
free(item->value);
free(item);
}

void free_table(HashTable *table)
{
// Frees the table
for (int i = 0; i < table->size; i++)
{
```

```
Ht_item *item = table->items[i];
if (item != NULL)
free_item(item);
}

free(table->items);
free(table);
}

void handle_collision(HashTable *table, unsigned long index, Ht_item
*item)
{
}

void ht_insert(HashTable *table, char *key, char *value)
{
// Create the item
Ht_item *item = create_item(key, value);

// Compute the index
unsigned long index = hash_function(key);

Ht_item *current_item = table->items[index];

if (current_item == NULL)
{
// Key does not exist.
if (table->count == table->size)
{
// Hash Table Full
printf("Insert Error: Hash Table is full\n");
// Remove the create item
free_item(item);
return;
}

// Insert directly
table->items[index] = item;
```

```c
table->count++;
}

else
{
// Scenario 1: We only need to update value
if (strcmp(current_item->key, key) == 0)
{
strcpy(table->items[index]->value, value);
return;
}

else
{
// Scenario 2: Collision
// We will handle case this a bit later
handle_collision(table, index, item);
return;
}
}
}

char *ht_search(HashTable *table, char *key)
{
// Searches the key in the hashtable
// and returns NULL if it doesn't exist
int index = hash_function(key);
Ht_item *item = table->items[index];

// Ensure that we move to a non NULL item
if (item != NULL)
{
if (strcmp(item->key, key) == 0)
return item->value;
}
return NULL;
}
```

```
void print_search(HashTable *table, char *key)
{
char *val;
if ((val = ht_search(table, key)) == NULL)
{
printf("Key:%s does not exist\n", key);
return;
}
else
{
printf("Key:%s, Value:%s\n", key, val);
}
}

void print_table(HashTable *table)
{
printf("\nHash Table\n-------------------\n");
for (int i = 0; i < table->size; i++)
{
if (table->items[i])
{
printf("Index:%d, Key:%s, Value:%s\n", i, table->items[i]->key, table-
>items[i]->value);
}
}
printf("-------------------\n\n");
}

int main()
{
HashTable *ht = create_table(CAPACITY);
ht_insert(ht, "1", "First address");
ht_insert(ht, "2", "Second address");
print_search(ht, "1");
print_search(ht, "2");
print_search(ht, "3");
print_table(ht);
free_table(ht);
return 0;
}
```

OUTPUT :



```
linuxcode@linuxcode:~/190905514/FIFTH-SEM/CD-LAB/LAB1$ gcc pgm4.c -o pgm4
linuxcode@linuxcode:~/190905514/FIFTH-SEM/CD-LAB/LAB1$ ./pgm4
Key:1, Value:First address
Key:2, Value:Second address
Key:3 does not exist

Hash Table
-------------------
Index:49, Key:1, Value:First address
Index:50, Key:2, Value:Second address
-------------------

linuxcode@linuxcode:~/190905514/FIFTH-SEM/CD-LAB/LAB1$ 
```