NAME: Tejas R **SUBJECT CODE: UE19CS152**

SRN: PES2UG19CS429 **SECTION: D**

QUESTION

17. Bug tracking software

This program should help a user file a bug (with all details – a unique ID which is generated by the program, Type of the bug, a brief description, Priority for the bug, time at which it was filed, status of the bug ("Not yet assigned", "In process", "Fixed", "Delivered", name of the person who filed the bug, etc). There should be a provision to change the status of the bug, get a report on bug statistics) all of these using file operations.

Any software has bugs. This program is to help users to report bugs,

User (A customer) would key in the following details:

A brief description

Name of the user who is filing this bug report

Type of the bug – Major, Minor, Cosmetic

Priority for the bug – Low, Medium, High

Status of the bug – Not assigned

Another kind of user (Manager) would look at these bugs from the same file and assign it to a person to fix it. This person should be able to change the status of the bug to one of (Assigned, Being fixed, Fixed, Delivered)

At any point in time, one should be able to get the list of all bugs:

1. Filed by the same person

2. Have the same category

3. Have the same status

CLIENT.C:

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<time.h>
#include"server.h"
int main () // main function that controls all other functions
{
  int checkid,no,ch; char peras[50]; char changestto[50]; char search[50];
   do
   {
 printf("Menu\n");
   printf("1. Add a bug (maximum 100)\n");
   printf("2. Display all bugs and total number of bugs\n");
   printf("3. Assign bug to a person (manager required) \n");
   printf("4. Get list of bugs filed by same person\n");
   printf("5. Get list of bugs having same priority \n");
   printf("6. Get list of all bugs having same status\n");
   printf("7. Get list of all bugs having same type\n");
   printf("8. Change status of bug (only assigned person can do so)\n");
   printf("9. Exit\n");
   printf("Enter choice\n");
   scanf("%d",&ch);
   switch(ch) // to call the functions depending on need of the user
   {
      case 1:
```

```
add();
break;
case 2:
printf("Enter 0 to display all bugs, otherwise enter specific bug ID for its details\n");
scanf("%d",&checkid);
no=disp(checkid);
printf("Total number of bugs= %d\n",no);
break;
case 3:
// taking in details, using %[^\n] to change delimiter of scanf for strings from space to newline
printf("Enter ID of bug\n");
scanf("%d",&checkid);
printf("Enter name of person to who bug is assigned (max 50 letters)\n");
scanf(" %[^\n]",peras);
assign(peras,checkid);
break;
case 4:
// taking in details, using %[^\n] to change delimiter of scanf for strings from space to newline
printf("Enter name of person to be searched\n");
scanf(" %[^\n]",search);
no=person(search);
printf("Total number of bugs filed by %s= %d\n",search,no); //displaying number of bugs filed by specific person
break;
case 5:
```

```
// taking in details, using %[^\n] to change delimiter of scanf for strings from space to newline

printf("Enter priority of bug to be searched (low, medium, high)\n");

scanf(" %[^\n]",search);

no=pri(search);

printf("Total number of bugs filed with %s priority= %d\n",search,no); //displaying number of bugs filed with specific priority

break;

case 6:

// taking in details, using %[^\n] to change delimiter of scanf for strings from space to newline

printf("Enter status of bug to be searched (not assigned, assigned, being fixed, fixed, delivered)\n");

scanf(" %[^\n]",search);

no=status(search);

printf("Total number of bugs filed with %s status= %d\n",search,no); //displaying number of bugs filed with specific status

break;

case 7:

// taking in details, using %[^\n] to change delimiter of scanf for strings from space to newline

printf("Enter type of bug to be searched (major, minor, cosmetic) \n");

scanf(" %[^\n]",search);

no=type(search);

printf("Total number of bugs filed with %s type= %d\n",search,no); //displaying number of bugs filed with specific type

break;

case 8:
```

```
        // taking in details, using %[^\n] to change delimiter of scanf for strings from space to
newline
        printf("Enter ID of bug\n");
        scanf("%d",&checkid);
        printf("Enter your name (person assigned to bug)\n");
        scanf(" %[^\n]",peras);
        printf("Enter the changed status\n");
        scanf(" %[^\n]",changestto);
        changestat(checkid,peras,changestto);
        break;
        case 9:
        exit(0); //EXIT_SUCCESS;
        // allows the user to exit the program at any time he chooses
        default: // for wrong input
        printf("Invalid entry\n");
    } // keeps calling the menu until user is done with his tasks
   }while(1);
}
```

SERVER.C

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<time.h>
#include"server.h"
int checkID() //function to find the last assigned ID
```

```
{
   FILE *check;
   struct bug ID;
   ID.id=0; // default ID in case file is empty
   check= fopen ("bug", "rb"); // Open file for reading
   if (check == NULL)
   return 0;
   // read file contents till end of file
   while(fread(&ID, sizeof(struct bug), 1, check)>0) ;
   return ID.id;
}
void datime(char *date) // function to get system date and time
{
   time_t rawtime;
   struct tm*info;
   time(&rawtime);
   info=localtime(&rawtime);
   strcpy(date,asctime(info));
}
void add() //function to add a bug to the file
{ struct bug re;int c=0;
// taking in details, using %[^\n] to change delimiter of scanf for strings from space to
newline
  printf("Enter name of user filing bug (max 50 letters)\n");
  scanf(" %[^\n]",re.name);
  printf("Enter details of bug in less than 200 letters\n");
```

```
 scanf(" %[^\n]",re.details);
 printf("Enter type of the bug (major,minor or cosmetic)\n");
 scanf("%s",re.type);
 printf("Enter priority of bug (low, medium or high) \n");
 scanf("%s",re.prio);
 strcpy(re.stat,"Not assigned"); // automatic assignment of bug status upon filing
 FILE *outfile;
  outfile = fopen ("bug", "ab"); // open file to append
  if (outfile == NULL)
  {
     fprintf(stderr, "\nError opening file\n");
     exit (1);
  } //handling error
  c=checkID(); //to automatically assign bug ID and ensure limit of 100 is maintained
if(c>99)
{
  printf("Maximum limit of 100 reached\n");
  return;
}
  re.id=c+1;   // automatically assign ID
  strcpy(re.assiper," "); //blank space for assigned person which needs to be decided by
manager
  datime(re.time);
  fwrite (&re, sizeof(struct bug), 1, outfile);  // write structure to file
  if(fwrite != 0)
     {
```

```
        printf("contents to file written successfully !\n");
        printf("Your bug ID is %d \n",re.id);
      }
   else
      printf("error writing file !\n"); //handling error
   fclose (outfile);  // close file
}
int disp(int choice) //function to display contents of file
{int i=0,nf=0;
FILE *infile;
   struct bug in;
   infile = fopen ("bug", "rb");  // Open file for reading
   if (infile == NULL)
   {
      fprintf(stderr, "\nError opening file\n");
      exit (1);
   }
   if(choice==0)
  {  // read file contents till end of file
     while(fread(&in, sizeof(struct bug), 1, infile)>0)
      printf ("id = %d    name = %s   details=%s  date,time=%s type=%s     priority=%s
status=%s   assigned person=%s\n",
      in.id, in.name, in.details, in.time, in.type, in.prio, in.stat,in.assiper); //print details of
bug on screen
  }
  if(choice!=0)
  {
```

```
        // read file contents till end of file
    while(fread(&in, sizeof(struct bug), 1, infile)>0)
    {
       if (in.id==choice)  //check for ID match
        printf ("id = %d    name = %s   details=%s  date,time=%s type=%s     priority=%s
status=%s   assigned person=%s\n",
        in.id, in.name, in.details, in.time, in.type, in.prio, in.stat,in.assiper); //print details of
bug on screen
       else
       nf++;
     i++;
    }
    if(nf==i) //check if no ID matched input ID
    printf("Matching ID not found\n");
   }
    fclose (infile);  // close file
    return checkID();
}
void assign(char as[50],int sid) // function for manager to assign a person to specific bug
{  int i=0,nf=0,j=1;
struct bug assi[100];
FILE *infile;
infile = fopen ("bug", "rb"); // open file for reading
    if (infile == NULL)
    {
        fprintf(stderr, "\nError opening file\n");
        exit (1);
```

```
   } // handling error
   // read file contents till end of file
   while(fread(&assi[i], sizeof(struct bug), 1, infile)>0)
   {
     if (assi[i].id==sid)  //check for ID match
     {
     strcpy(assi[i].assiper,as); //filling in the name according to manager input
     strcpy(assi[i].stat,"Assigned"); // automatically change status after person has been assigned to specific bug by manager
     }
     else
     nf++;
    i++;
   }
   if(nf==i) //check if no ID matched input ID
   {
   printf("Matching ID not found\n");
   return; // if input ID does not match, no need to rewrite the file so exit the function
   }
   fclose(infile); // close file
  FILE *p;
p=fopen("bug","wb");  //open file for writing, erase the previously stored data and rewrite the file using the new structure
if ((p == NULL))
   {
      fprintf(stderr, "\nError opening file\n");
      exit (1);
```

```
   } //handling error
fwrite(&assi[0],sizeof(struct bug),1,p); // write the first bug details
fclose(p); // close file
FILE *change;
change=fopen("bug","ab"); // open file to append for the other bug details to avoid rewriting the file
if((change==NULL))
{
   fprintf(stderr,"\nError opening file\n");
   exit(1);
} //handling error
 while(j<i)
 {
    fwrite(&assi[j], sizeof(struct bug), 1, change);  //writing the whole new structure into the file
    j++;
 }
   if(fwrite != 0)
      {
         printf("contents to file written and status automatically changed to assigned\n");
      }
   else
      printf("error writing file !\n");//handling error
fclose(change); // close file
}
void changestat(int nid, char cper[50],char chan[50]) //function for assigned person of a specific bug to change its status
```

```
{ int i=0,nfc=0,j=1;
struct bug n[100];
FILE *fp;
   fp = fopen ("bug", "rb"); // Open file for reading
   if ((fp == NULL))
   {
      fprintf(stderr, "\nError opening file\n");
      exit (1);
   } // handling error
   // read file contents till end of file
   while(fread(&n[i], sizeof(struct bug), 1, fp)>0)
     {
     if (n[i].id==nid) // checking for input ID in the file
     {printf("Matching ID found\n");
     if(strcmpi(n[i].assiper,cper)==0) // checking for input name of assigned person within
the file
        strcpy(n[i].stat,chan); //changing status if both ID and name match
      else
      {
      printf("But assigned person name does not match\n");
      return; // if input assigned person name does not match, no need to rewrite file so exit
the function
      }
      }
      else
       nfc++;
     i++;
```

```
      }
      if(nfc==i) //check if no ID matched input ID
      {printf("Matching ID not found\n");
      return; //if input ID does not match no need to rewrite file so exit the function
      }
 fclose (fp); // close file
FILE *p;
p=fopen("bug","wb");  //open file for writing, erase the previously stored data and rewrite the file using the new structure
if ((p == NULL))
   {
      fprintf(stderr, "\nError opening file\n");
      exit (1);
   } // handling error
fwrite(&n[0],sizeof(struct bug),1,p); //writing the first bug details
fclose(p); //close file
FILE *change;
change=fopen("bug","ab"); //open file to append for the other bug details to avoid rewriting the file
if((change==NULL))
{
   fprintf(stderr,"\nError opening file\n");
   exit(1);
}// handling error
 while(j<i)
 {
```

```
    fwrite(&n[j], sizeof(struct bug), 1, change);  //writing the whole new structure into the
file
    j++;
 }
   if(fwrite != 0)
      {
         printf("contents to file written successfully !\n");
      }
   else
      printf("error writing file !\n"); //handling error
fclose(change); // close file
}
int person(char per[50]) //function to search for bugs filed by a specific person
{  struct bug p;int count=0;

  FILE *fp;
   fp = fopen ("bug", "rb"); // Open file for reading
   if ((fp == NULL))
   {
      fprintf(stderr, "\nError opening file\n");
      exit (1);
   } //handling error
   // read file contents till end of file
   while(fread(&p, sizeof(struct bug), 1, fp)>0)
     {
     if (strcmpi(per,p.name)==0) //checking if input name matches any user name within the
file
```

```
      {
       printf ("id = %d    name = %s   details=%s  date,time=%s type=%s     priority=%s
status=%s   assigned person=%s\n",
       p.id, p.name, p.details, p.time, p.type, p.prio, p.stat,p.assiper); //printing details of bug
filed by input user name
       count++;
      }
    }
  fclose(fp); //close file
  return count;
}
int pri(char pr[50]) //function search for bugs with specific priority
{  struct bug p;int count=0;
FILE *fp;
   fp = fopen ("bug", "rb"); // Open file for reading
   if ((fp == NULL))
   {
      fprintf(stderr, "\nError opening file\n");
      exit (1);
   } // handling error
   // read file contents till end of file
   while(fread(&p, sizeof(struct bug), 1, fp)>0)
     {
     if (strcmpi(pr,p.prio)==0) //checking if input priority matches with any within the file
     {
     printf ("id = %d    name = %s   details=%s  date,time=%s type=%s     priority=%s
status=%s   assigned person=%s\n",
```

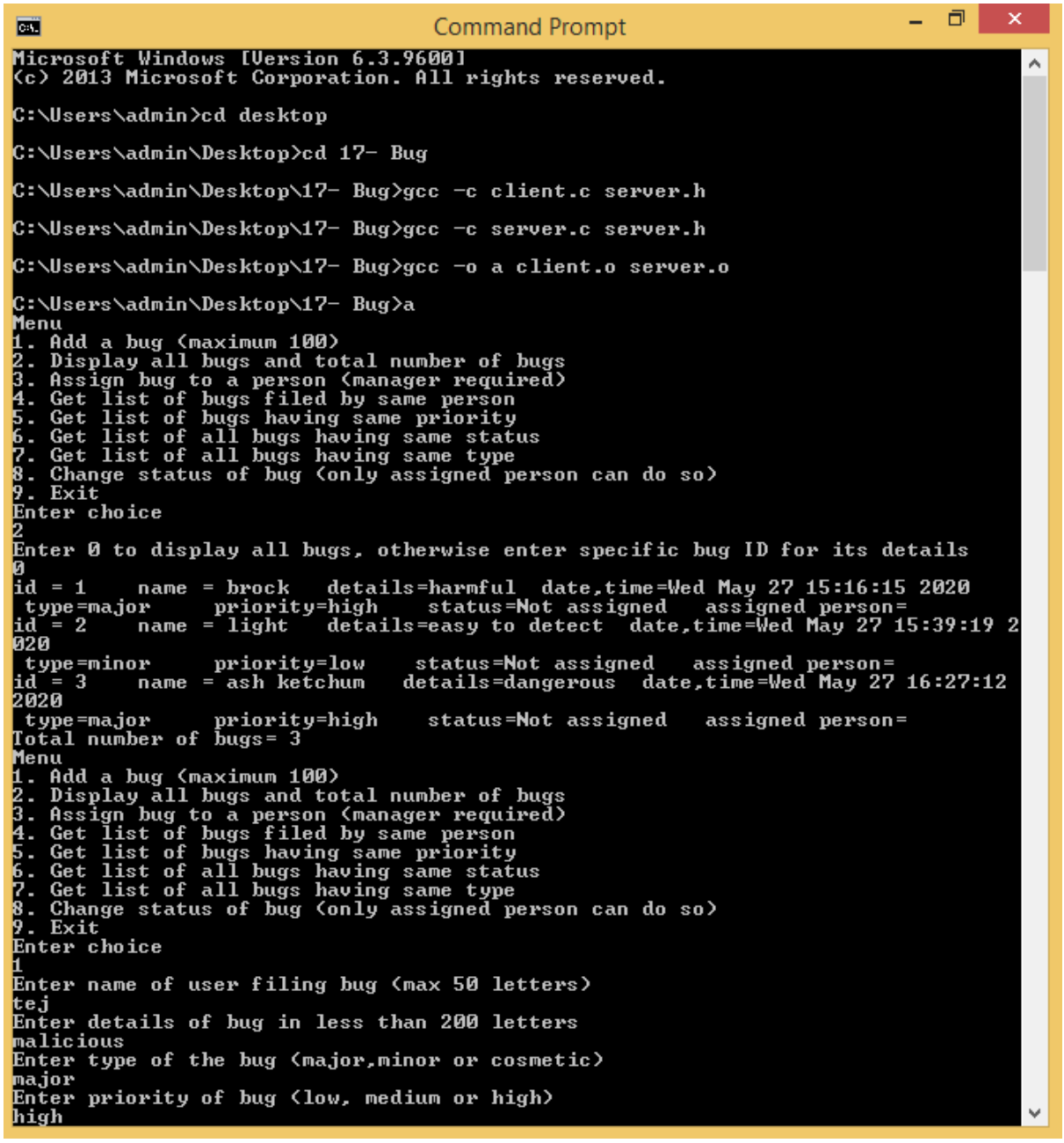
```
      p.id, p.name, p.details, p.time, p.type, p.prio, p.stat,p.assiper); // printing details of bug if priority matches
      count++;
     }
    }
fclose(fp); //close file
return count;
}
int status(char st[50]) // function to search for bugs with specific status
{  struct bug p;int count=0;
  FILE *fp;
   fp = fopen ("bug", "rb"); // Open file for reading
   if ((fp == NULL))
   {
      fprintf(stderr, "\nError opening file\n");
      exit (1);
   } // handling error
   // read file contents till end of file
   while(fread(&p, sizeof(struct bug), 1, fp)>0)
     {
     if (strcmpi(st,p.stat)==0) //checking if input status matches withy any within the file
     {
     printf ("id = %d    name = %s   details=%s  date,time=%s type=%s     priority=%s status=%s   assigned person=%s\n",
      p.id, p.name, p.details, p.time, p.type, p.prio, p.stat,p.assiper); // printing details of bug if status matches
      count++;
```

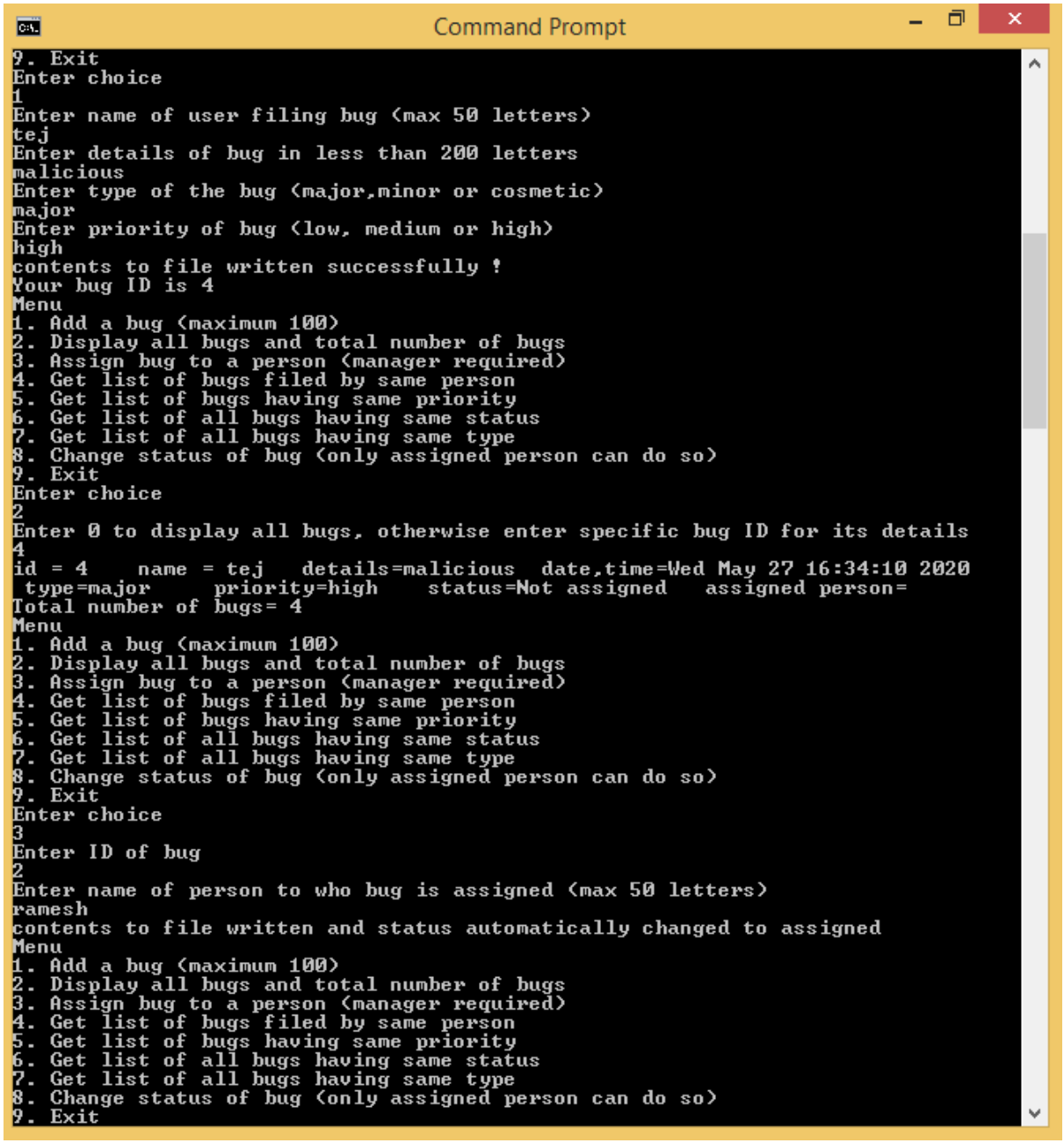
```
      }
     }
  fclose(fp);// close file
  return count;
}
int type(char ty[50]) // function to search for bugs with a specific type
{ struct bug p;int count=0;
   FILE *fp;
    fp = fopen ("bug", "rb");  // Open file for reading
    if ((fp == NULL))
    {
       fprintf(stderr, "\nError opening file\n");
       exit (1);
    } // handling error
    // read file contents till end of file
    while(fread(&p, sizeof(struct bug), 1, fp)>0)
      {
      if (strcmpi(ty,p.type)==0) // checking if input type matches with any within the file
      {
        printf ("id = %d    name = %s   details=%s  date,time=%s type=%s     priority=%s
status=%s   assigned person=%s\n",
       p.id, p.name, p.details, p.time, p.type, p.prio, p.stat,p.assiper); // printing details of bug
if type matches
       count++;
      }
     }
  fclose(fp); //close file
```

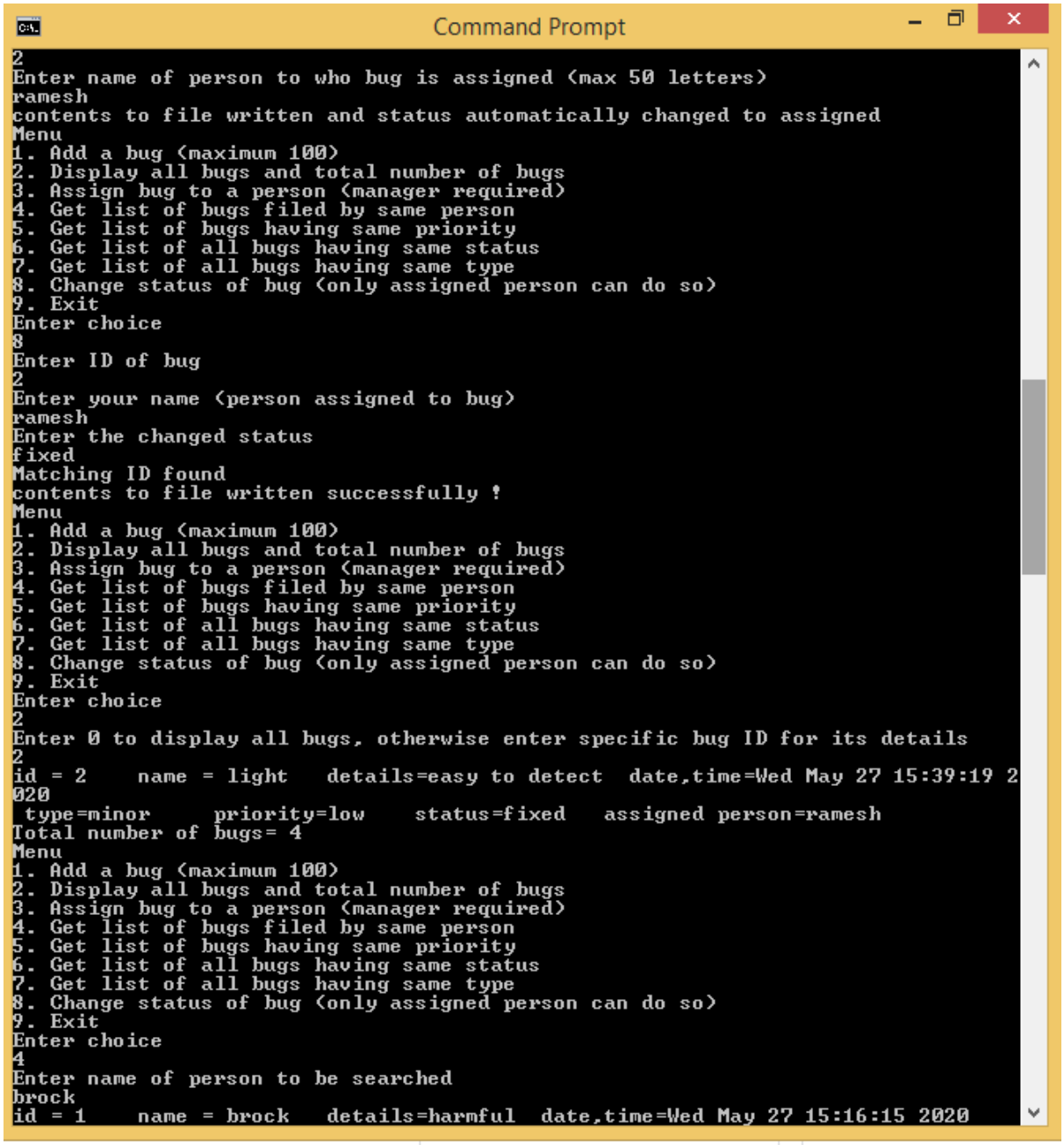
```
  return count;
}
```

SERVER.H

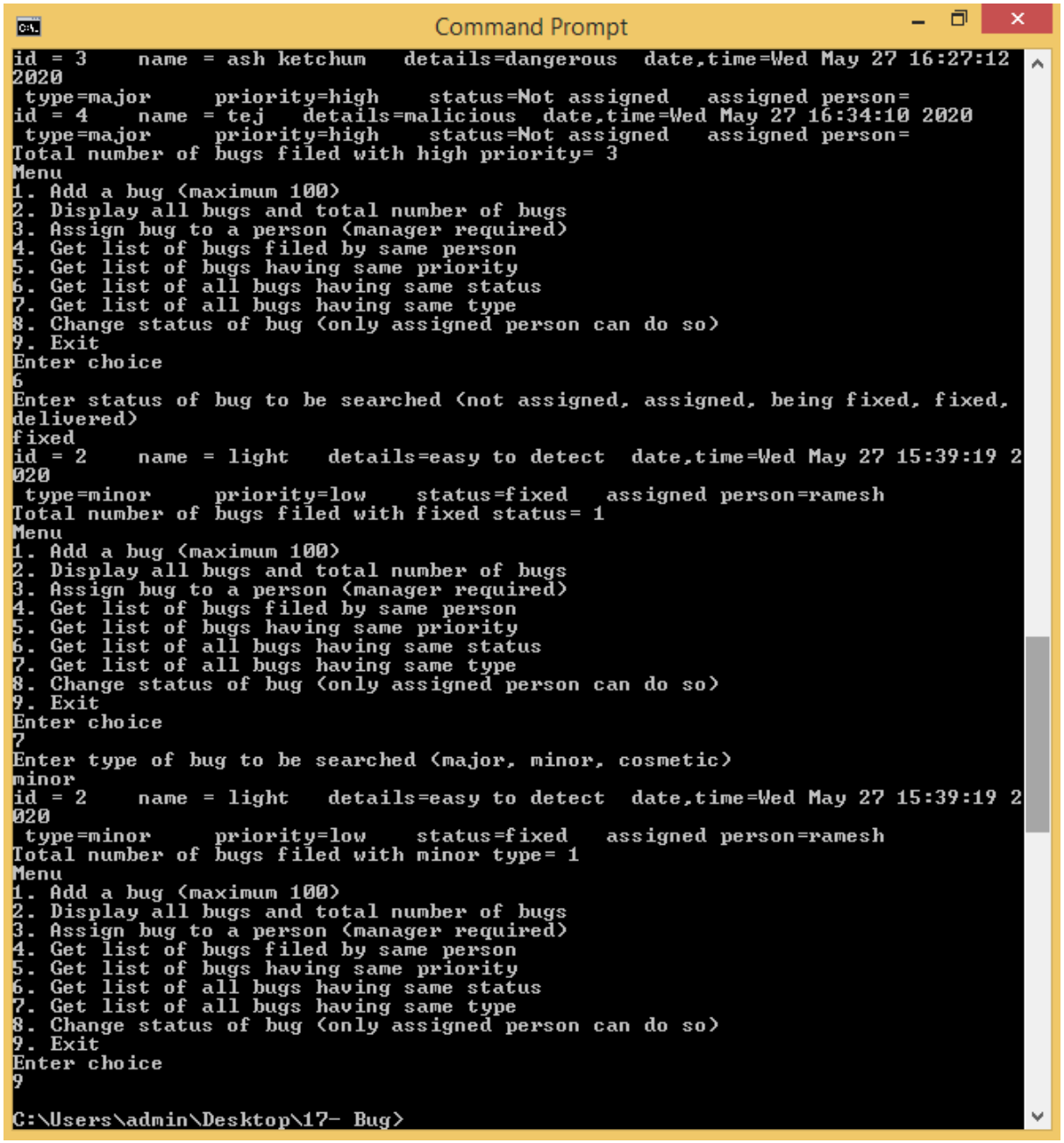
```
struct bug // a structure to read and write
{
   int id; //stores bug ID
   char name[50]; //stores name of user filing bug
   char details[200];  //stores details of bug
   char type[50]; //stores type of bug
   char prio[50]; // stores priority of bug
   char stat[50]; //stores status of bug
   char assiper[50]; //stores name of person assigned to bug
   char time[50]; //stores time and date when bug was filed
};
int checkID();
void datime(char *);
void add();
int disp(int);
void assign(char[],int);
void changestat(int, char[],char[]);
int person(char[]);
int pri(char[]);
int status(char[]);
int type(char[]);
```


Command Prompt
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.
C:\Users\admin>cd desktop
C:\Users\admin\Desktop>cd 17- Bug
C:\Users\admin\Desktop\17- Bug>gcc -c client.c server.h
C:\Users\admin\Desktop\17- Bug>gcc -c server.c server.h
C:\Users\admin\Desktop\17- Bug>gcc -o a client.o server.o
C:\Users\admin\Desktop\17- Bug>a
Menu
1. Add a bug (maximum 100)
2. Display all bugs and total number of bugs
3. Assign bug to a person (manager required)
4. Get list of bugs filed by same person
5. Get list of bugs having same priority
6. Get list of all bugs having same status
7. Get list of all bugs having same type
8. Change status of bug (only assigned person can do so)
9. Exit
Enter choice
2
Enter 0 to display all bugs, otherwise enter specific bug ID for its details
0
id = 1 name = brock details=harmful date,time=Wed May 27 15:16:15 2020
type=major priority=high status=Not assigned assigned person=
id = 2 name = light details=easy to detect date,time=Wed May 27 15:39:19 2020
type=minor priority=low status=Not assigned assigned person=
id = 3 name = ash ketchum details=dangerous date,time=Wed May 27 16:27:12 2020
type=major priority=high status=Not assigned assigned person=
Total number of bugs= 3
Menu
1. Add a bug (maximum 100)
2. Display all bugs and total number of bugs
3. Assign bug to a person (manager required)
4. Get list of bugs filed by same person
5. Get list of bugs having same priority
6. Get list of all bugs having same status
7. Get list of all bugs having same type
8. Change status of bug (only assigned person can do so)
9. Exit
Enter choice
1
Enter name of user filing bug (max 50 letters)
tej
Enter details of bug in less than 200 letters
malicious
Enter type of the bug (major,minor or cosmetic)
major
Enter priority of bug (low, medium or high)
high


Command Prompt
9. Exit
Enter choice
1
Enter name of user filing bug (max 50 letters)
tej
Enter details of bug in less than 200 letters
malicious
Enter type of the bug (major,minor or cosmetic)
major
Enter priority of bug (low, medium or high)
high
contents to file written successfully !
Your bug ID is 4
Menu
1. Add a bug (maximum 100)
2. Display all bugs and total number of bugs
3. Assign bug to a person (manager required)
4. Get list of bugs filed by same person
5. Get list of bugs having same priority
6. Get list of all bugs having same status
7. Get list of all bugs having same type
8. Change status of bug (only assigned person can do so)
9. Exit
Enter choice
2
Enter 0 to display all bugs, otherwise enter specific bug ID for its details
4
id = 4 name = tej details=malicious date,time=Wed May 27 16:34:10 2020
type=major priority=high status=Not assigned assigned person=
Total number of bugs= 4
Menu
1. Add a bug (maximum 100)
2. Display all bugs and total number of bugs
3. Assign bug to a person (manager required)
4. Get list of bugs filed by same person
5. Get list of bugs having same priority
6. Get list of all bugs having same status
7. Get list of all bugs having same type
8. Change status of bug (only assigned person can do so)
9. Exit
Enter choice
3
Enter ID of bug
2
Enter name of person to who bug is assigned (max 50 letters)
ramesh
contents to file written and status automatically changed to assigned
Menu
1. Add a bug (maximum 100)
2. Display all bugs and total number of bugs
3. Assign bug to a person (manager required)
4. Get list of bugs filed by same person
5. Get list of bugs having same priority
6. Get list of all bugs having same status
7. Get list of all bugs having same type
8. Change status of bug (only assigned person can do so)
9. Exit


Command Prompt
2
Enter name of person to who bug is assigned (max 50 letters)
ramesh
contents to file written and status automatically changed to assigned
Menu
1. Add a bug (maximum 100)
2. Display all bugs and total number of bugs
3. Assign bug to a person (manager required)
4. Get list of bugs filed by same person
5. Get list of bugs having same priority
6. Get list of all bugs having same status
7. Get list of all bugs having same type
8. Change status of bug (only assigned person can do so)
9. Exit
Enter choice
8
Enter ID of bug
2
Enter your name (person assigned to bug)
ramesh
Enter the changed status
fixed
Matching ID found
contents to file written successfully !
Menu
1. Add a bug (maximum 100)
2. Display all bugs and total number of bugs
3. Assign bug to a person (manager required)
4. Get list of bugs filed by same person
5. Get list of bugs having same priority
6. Get list of all bugs having same status
7. Get list of all bugs having same type
8. Change status of bug (only assigned person can do so)
9. Exit
Enter choice
2
Enter 0 to display all bugs, otherwise enter specific bug ID for its details
2
id = 2 name = light details=easy to detect date,time=Wed May 27 15:39:19 2
020
type=minor priority=low status=fixed assigned person=ramesh
Total number of bugs= 4
Menu
1. Add a bug (maximum 100)
2. Display all bugs and total number of bugs
3. Assign bug to a person (manager required)
4. Get list of bugs filed by same person
5. Get list of bugs having same priority
6. Get list of all bugs having same status
7. Get list of all bugs having same type
8. Change status of bug (only assigned person can do so)
9. Exit
Enter choice
4
Enter name of person to be searched
brock
id = 1 name = brock details=harmful date,time=Wed May 27 15:16:15 2020


Command Prompt
Enter name of person to be searched
brock
id = 1 name = brock details=harmful date,time=Wed May 27 15:16:15 2020
type=major priority=high status=Not assigned assigned person=
Total number of bugs filed by brock= 1
Menu
1. Add a bug (maximum 100)
2. Display all bugs and total number of bugs
3. Assign bug to a person (manager required)
4. Get list of bugs filed by same person
5. Get list of bugs having same priority
6. Get list of all bugs having same status
7. Get list of all bugs having same type
8. Change status of bug (only assigned person can do so)
9. Exit
Enter choice
5
Enter priority of bug to be searched (low, medium, high)
high
id = 1 name = brock details=harmful date,time=Wed May 27 15:16:15 2020
type=major priority=high status=Not assigned assigned person=
id = 3 name = ash ketchum details=dangerous date,time=Wed May 27 16:27:12
2020
type=major priority=high status=Not assigned assigned person=
id = 4 name = tej details=malicious date,time=Wed May 27 16:34:10 2020
type=major priority=high status=Not assigned assigned person=
Total number of bugs filed with high priority= 3
Menu
1. Add a bug (maximum 100)
2. Display all bugs and total number of bugs
3. Assign bug to a person (manager required)
4. Get list of bugs filed by same person
5. Get list of bugs having same priority
6. Get list of all bugs having same status
7. Get list of all bugs having same type
8. Change status of bug (only assigned person can do so)
9. Exit
Enter choice
6
Enter status of bug to be searched (not assigned, assigned, being fixed, fixed,
delivered)
fixed
id = 2 name = light details=easy to detect date,time=Wed May 27 15:39:19 2
020
type=minor priority=low status=fixed assigned person=ramesh
Total number of bugs filed with fixed status= 1
Menu
1. Add a bug (maximum 100)
2. Display all bugs and total number of bugs
3. Assign bug to a person (manager required)
4. Get list of bugs filed by same person
5. Get list of bugs having same priority
6. Get list of all bugs having same status
7. Get list of all bugs having same type
8. Change status of bug (only assigned person can do so)
9. Exit
Enter choice



ASSUMPTIONS:

- User can enter a maximum of 100 bugs.
- While entering input the user must not make any accidental mistakes and data entered must match the type of data that the system expects.
- The user must know the unique ID of the bug for various tasks. He can find the ID through other details as well.