

Lab Assignment-3

Name: Sadman Sharif

ID: 20101107

Section: 02

Course: CSE321

Submission Date: 24/11/2022

Function & Struct

=====Task-1=====

```
#include <stdio.h>

struct breakfast
{
    float price;
    float quantity;
};

int main()
{
    float Buy,total,Person;
    struct breakfast paratha,water,vegi;
    printf("Quantity Of Paratha: ");
    scanf("%f", &paratha.quantity);
    printf("Unit Price: ");
    scanf("%f", &paratha.price);
    printf("Quantity Of vegetable: ");
    scanf("%f", &vegi.quantity);
    printf("Unit Price: ");
    scanf("%f", &vegi.price);
    printf("Quantity Of Mineral Water: ");
    scanf("%f", &water.quantity);
    printf("Unit Price: ");
    scanf("%f", &water.price);
    printf("Number of people: ");
    scanf("%f", & Buy);

    total = (paratha.quantity*paratha.price)+(water.quantity*
water.price)+(vegi.quantity *vegi.price);

    Person=total/Buy;

    printf("Individual people will pay: %f\n", Person);

    return 0;
}
```

Task-1 (Output)

```
ami.c:1:1: note: include '<stdio.h>' or provide a declaration of 'printf'
+++ |+#include <stdio.h>
1 | struct breakfast
ami.c:11:5: warning: incompatible implicit declaration of built-in function 'printf' [-Wbuiltin-declaration-mismatch]
11 |     printf("Quantity Of Paratha: ");
    |     ^~~~~~
ami.c:11:5: note: include '<stdio.h>' or provide a declaration of 'printf'
ami.c:12:5: warning: implicit declaration of function 'scanf' [-Wimplicit-function-declaration]
12 |     scanf("%f", &paratha.quantity);
    |     ^~~~~~
ami.c:12:5: note: include '<stdio.h>' or provide a declaration of 'scanf'
ami.c:12:5: warning: incompatible implicit declaration of built-in function 'scanf' [-Wbuiltin-declaration-mismatch]
ami.c:12:5: note: include '<stdio.h>' or provide a declaration of 'scanf'
sadman@sadman:~/Desktop$ gcc ami.c
sadman@sadman:~/Desktop$ ./a.out
Quantity Of Paratha: 25
Unit Price: 10
Quantity Of vegetable: 5
Unit Price: 20
Quantity Of Mineral Water: 20
Unit Price: 20
Number of people: 6
Individual people will pay: 125.000000
sadman@sadman:~/Desktop$
```

=====Task-2=====

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

int perfect(int n){
    int s=0;
    for(int i= 1; i<n; i++){
        if (n% i == 0){
            s+=i;
        }
    }
    if(s==n){
        return n;
    }
    else{return 0;}
}

int main(){

    int l,u,r;
    scanf("%d",&l);
    scanf("%d",&u);

    for(int j=1; j<=u; j++){
        r=perfect(j);
        if (r!=0){
            printf("%d\n",r);
        }
    }
}
```

Task-2 (Output)

```
sadman@sadman:~/Desktop$ gcc ami.c
sadman@sadman:~/Desktop$ ./a.out
1
10000
6
28
496
8128
sadman@sadman:~/Desktop$
```

```
=====
```

System Call

```
=====
```

=====Task-1=====

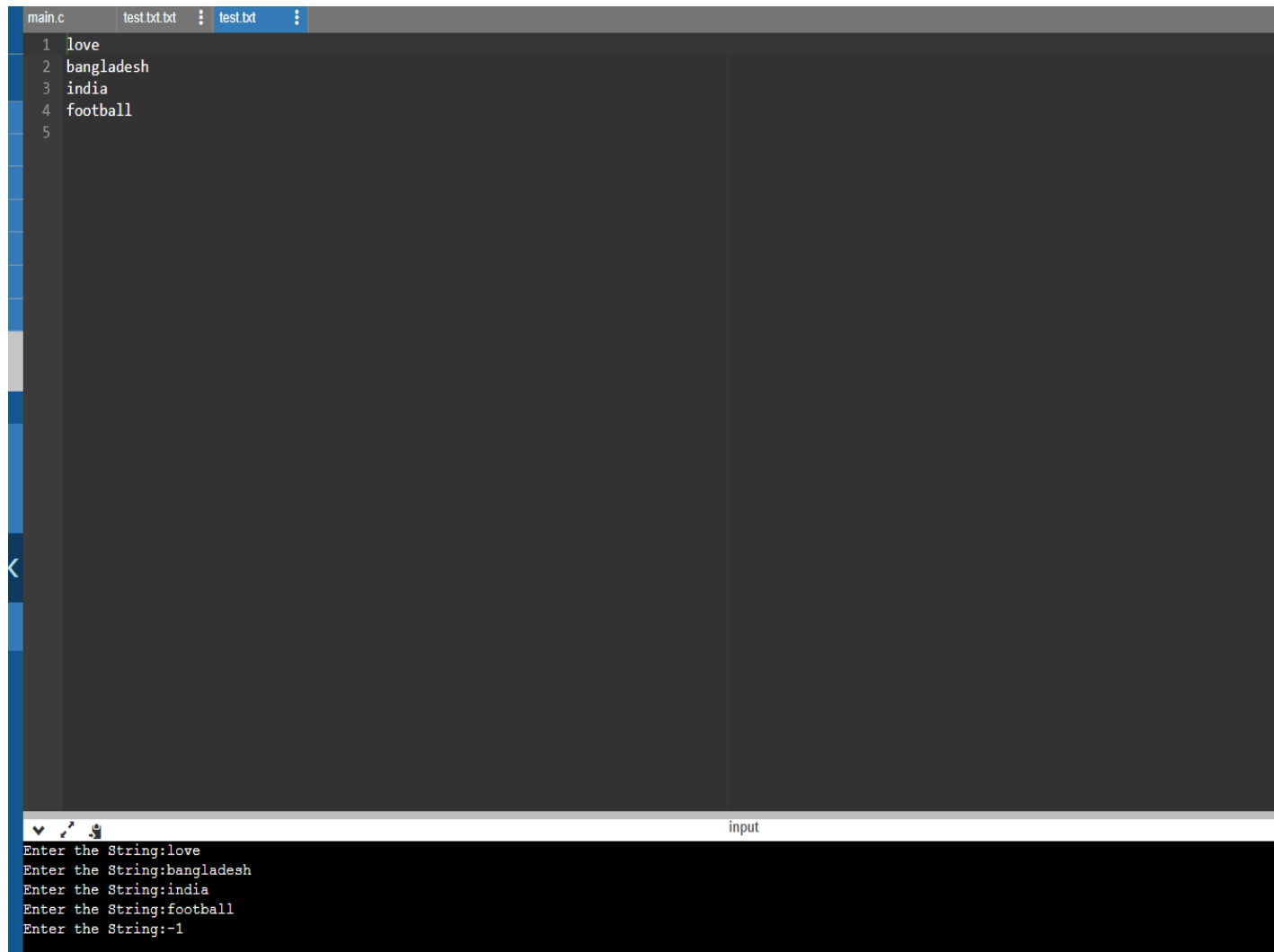
```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>

int main( int argc, char *argv[]){
    char name[50];
    char line[150];
    FILE *fp=fopen(argv[1],"w");

    while(1){
        char inp[100];
        printf("Enter the String:");
        scanf("%s",inp);

        int x=strcmp(inp,"-1");
        if(x==0){
            break;
        }
        fprintf(fp,"%s\n",inp);
    }
    fclose(fp) ;
    return 0;
}
```

Task-1 (Output)



The image shows a code editor window with a dark theme. The editor has two tabs: 'main.c' and 'test.txt'. The 'main.c' tab is active, showing a C program with five lines of code. The code is as follows:

```
1 love
2 bangladesh
3 india
4 football
5
```

Below the code editor is a terminal window. The terminal has a title bar with the word 'input' on the right. The terminal displays the output of the program, which consists of five lines of text, each preceded by the prompt 'Enter the String:':

```
Enter the String:love
Enter the String:bangladesh
Enter the String:india
Enter the String:football
Enter the String:-1
```

=====Task-2=====

```
#include <stdio.h>
#include <unistd.h>

int main(){
    int i=fork();
    if(i==0){

        int j=fork();
        if(j==0){
            printf("I am grandchild\n");
        }
        else if(j>0){
            sleep(2);
            printf("I am child\n");
        }
        else{
            printf("Not a valid call");
        }
    }
    else if(i>0){
        sleep(3);
        printf("I am parent\n");
    }
    else{
        printf("Not a valid call");
    }
    return 0;
}
```


Task-2 (Output)

```
Hasta la Vista  
sadman@sadman:~/Desktop$ gcc ami.c  
sadman@sadman:~/Desktop$ ./a.out  
Enter the String:shdushfusf  
Segmentation fault (core dumped)  
sadman@sadman:~/Desktop$ gcc ami.c  
sadman@sadman:~/Desktop$ ./a.out  
I am grandchild  
I am child  
I am parent  
sadman@sadman:~/Desktop$
```

=====Task-3=====

```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>

int main()
{
    int P=3;
    int x=fork();
    int y=fork();
    int z=fork();

    if(x%2!=0)
    {
        fork();
        P=P+1;
    }

    printf("\n The total Processes are: %d \n",P);
    return 0;
}
```

Task-3 (Output)

```
sadman@sadman:~$ cd Desktop
sadman@sadman:~/Desktop$ gcc ami.c
sadman@sadman:~/Desktop$ ./a.out

The number of Processes are: 4

The number of Processes are: 3

sadman@sadman:~/Desktop$ The number of Processes are: 4

The number of Processes are: 4

The number of Processes are: 3

The number of Processes are: 3

The number of Processes are: 3

The number of Processes are: 4

The number of Processes are: 4

The number of Processes are: 4

The number of Processes are: 4

The number of Processes are: 4

sadman@sadman:~/Desktop$
```

=====Task-4=====

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>

int main(){
    int num[8];
    for(int i=0; i<8; i++){
        printf("Enter the number: ");
        scanf("%d", &num[i]);
    }
    int ref;
    pid_t p;
    p=fork();

    if(p==0){
        for(int i=0; i<8; i++)
        {
            for(int j=i+1; j<8; j++){
                if(num[i]<num[j])
                {
                    ref=num[i];
                    num[i]=num[j];
                    num[j]=ref;
                }
            }
        }
    }
}
```

```
for(int i=0; i<8; i++)
{
    printf("%d\n", num[i]);
}

else if(p>0)
{
    for(int i=0; i<8; i++)
    {
        if(num[i]%2!=0)
        {
            printf("%d (The number user gave is odd)\n", num[i]);
        }

        else
        {
            printf("%d (The number user gave is even)\n", num[i]);
        }
    }
}

else
{
    perror("There is an invalid input");
}
}
```

Task-4 (Output)

```
sadman@sadman:~$ cd Desktop
sadman@sadman:~/Desktop$ gcc ami.c
sadman@sadman:~/Desktop$ ./a.out
Enter the number: 1
Enter the number: 2
Enter the number: 3
Enter the number: 4
Enter the number: 5
Enter the number: 6
Enter the number: 7
Enter the number: 8
1 (The number user gave is odd)
2 (The number user gave is even)
3 (The number user gave is odd)
4 (The number user gave is even)
5 (The number user gave is odd)
6 (The number user gave is even)
7 (The number user gave is odd)
8 (The number user gave is even)
8
7
6
5
4
3
2
1
sadman@sadman:~/Desktop$
```

=====Task-5=====

```
#include <stdio.h>
#include <unistd.h>

int main(){
    int p, gp1, gp2, gp3;
    p=fork();

    if(p==0){
        printf("Child PID: %d\n",getpid());
        int gp1=fork();
        if (gp1==0){
            printf("Grand Child PID: %d\n",getpid());
        }

        else if(gp1>0){
            gp2=fork();
            if(gp2==0){
                printf("Grand Child PID: %d\n",getpid());
            }
            else if(gp2>0){
                gp3=fork();
                if(gp3==0){
                    printf("Grand Child PID: %d",getpid());
                }
            }
        }
    }
}
```

```
    else if(p>0){  
        printf("Parent PID: %d\n",getpid());  
    }  
    else{  
        printf("Invalid call");  
    }  
    return 0;  
}
```

Task-5 (Output)

```
the number of processes are: 3  
^C  
sadman@sadman:~/Desktop$ gcc ami.c  
sadman@sadman:~/Desktop$ ./a.out  
Parent PID: 3802  
Child PID: 3803  
sadman@sadman:~/Desktop$ Grand Child PID: 3804  
Grand Child PID: 3805  
Grand Child PID: 3806
```



```
=====
```

Threading

```
=====
```

=====Task-1=====

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <pthread.h>

int x=1;

void* printout(void * arg){

    printf("thread-%d running\n",x);
    printf("thread-%d closed\n",x);

    x+=1;
}

int main()
{
    pthread_t newThread;
    for(int i =0; i< 5; i++)
    {
        pthread_create(&newThread, NULL, printout, NULL);
        pthread_join(newThread,NULL);
    }
    pthread_exit(NULL);
    return 0;
}
```

Task-1 (Output)

```
8128
sadman@sadman:~/Desktop$ gcc ami.c
sadman@sadman:~/Desktop$ ./a.out
thread-1 running
thread-1 closed
thread-2 running
thread-2 closed
thread-3 running
thread-3 closed
thread-4 running
thread-4 closed
thread-5 running
thread-5 closed
sadman@sadman:~/Desktop$
```

=====Task-2=====

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <pthread.h>

int h=0;
int g=1;
void* printout(void *arg){

    for(int i=1;i<=5;i++){
        printf("thread-%d prints %d\n",h,g);
        g+=1;
    }
    h+=1;
}

int main(){

    pthread_t newThread;

    for(int i=0; i<5; i++){
        pthread_create(&newThread, NULL, printout, NULL);
        pthread_join(newThread,NULL);

    }
    pthread_exit(NULL);
    return 0;
}
```

Task-2 (Output)

```
sadman@sadman:~/Desktop$ gcc ami.c
sadman@sadman:~/Desktop$ ./a.out
thread-0 prints 1
thread-0 prints 2
thread-0 prints 3
thread-0 prints 4
thread-0 prints 5
thread-1 prints 6
thread-1 prints 7
thread-1 prints 8
thread-1 prints 9
thread-1 prints 10
thread-2 prints 11
thread-2 prints 12
thread-2 prints 13
thread-2 prints 14
thread-2 prints 15
thread-3 prints 16
thread-3 prints 17
thread-3 prints 18
thread-3 prints 19
thread-3 prints 20
thread-4 prints 21
thread-4 prints 22
thread-4 prints 23
thread-4 prints 24
thread-4 prints 25
sadman@sadman:~/Desktop$
```

=====Task-3=====

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

int count=0;
int r[3];

void* ascii_f(void *arg)
{
    char name[30];
    printf("Enter name of user: ");
    scanf("%s",name);

    int ascii=0;
    int l=strlen(name);

    for(int i=0; i<l; i++)
    {
        int ch=name[i];
        ascii+=ch;
    }
    r[count]=ascii;
    count+=1;
}

int main()
{
    pthread_t newThread;

    for(int i=0; i<3; i++)
    {
        pthread_create(&newThread, NULL, ascii_f, NULL);
        pthread_join(newThread,NULL);
    }

    if(r[0]==r[1] && r[0]==r[2] && r[1]==r[2] )
    {
        printf("Youreka\n");
    }
}
```

```
    else if(r[0]==r[1] || r[1]==r[2] || r[0]==r[2] )
    {
        printf("Miracle\n");
    }

    else
    {
        printf("Hasta la vista\n");
    }

    pthread_exit(NULL);
    return 0;
}
```

Task-3 (Output)

```
thread-4 prints 23
thread-4 prints 24
thread-4 prints 25
sadman@sadman:~/Desktop$ gcc ami.c
sadman@sadman:~/Desktop$ ./a.out
Enter name of user: Youreka
Enter name of user: Miracle
Enter name of user: Hasta la vista
Hasta la vista
sadman@sadman:~/Desktop$
```