//Struct & Function

TASK-1:

#include <stdio.h>

#include <string.h>

struct foods

{

float price1, price2, price3, item1, item2, item3, member,total\_cost, result;

};

float individual\_payment()

{

struct foods payment;

printf("Quantity Of Paratha: ");

scanf("%f",&payment.item1);

printf("Unit Price: ");

scanf("%f",&payment.price1);

printf("Quantity Of Vegetable: ");

scanf("%f",&payment.item2);

printf("Unit Price: ");

scanf("%f",&payment.price2);

printf("Quantity Of Mineral Water: ");

scanf("%f",&payment.item3);

printf("Unit Price: ");

scanf("%f",&payment.price3);

printf("Number of People: ");

scanf("%f",&payment.member);

payment.result = (payment.item1\*payment.price1 + payment.item2\*payment.price2 + payment.item3\*payment.price3)/payment.member;

return payment.result;

}

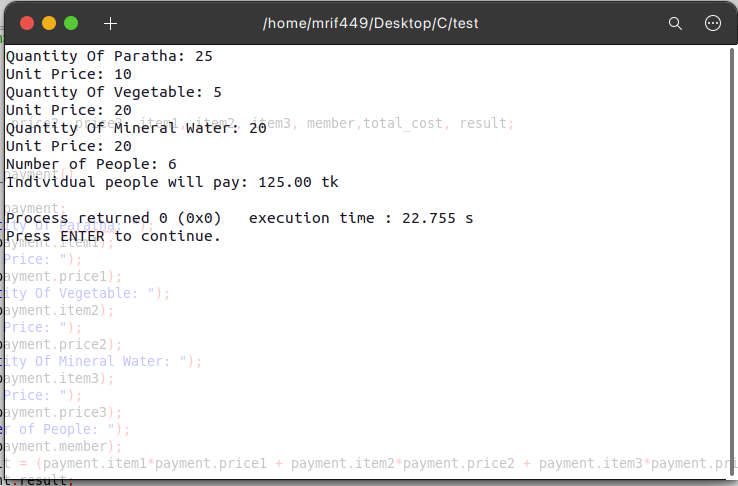
int main()

{

float x = individual\_payment();

printf("Individual people will pay: %.2f tk\n",x);

}



TASK-2:

#include <stdio.h>

#include <string.h>

struct perfect\_number

{

int x, start, end;

};

int number\_checking()

{

struct perfect\_number numbers;

scanf("%d",&numbers.start);

scanf("%d",&numbers.end);

for(numbers.x=numbers.start; numbers.x<=numbers.end; numbers.x++)

{

int value=numbers.x;

int i, sum = 0 ;

for(i = 1 ; i < value ; i++)

{

if(value % i == 0)

sum = sum + i ;

}

if (sum == value)

printf("%d\n\n", value);

}

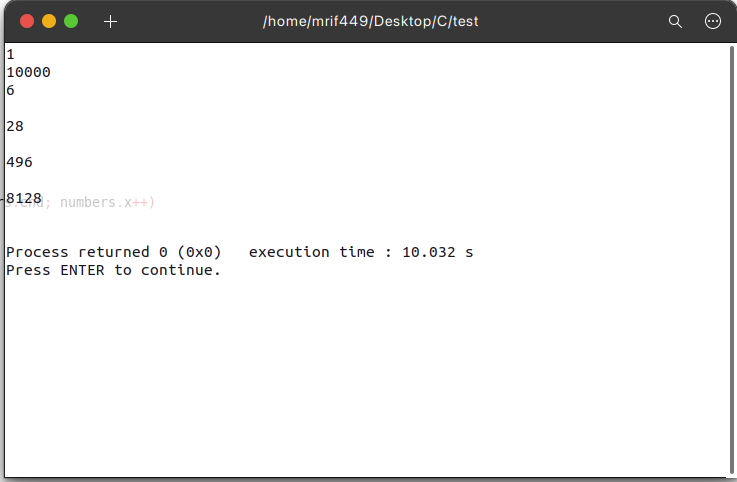
}

int main()

{

number\_checking();

}



//System Call

TASK-1:

#include<stdio.h>

#include<string.h>

#include<stdlib.h>

#include<fcntl.h>

#include<sys/stat.h>

#include<unistd.h>

/\* Sample Input:

./File /media/rahadul/D/C Diary

C

is

hard

-1

\*/

int main(int argc,const char \*\*argv)

{

int iFd = 0;

char \*Directory = NULL;

char \*Filename = NULL;

char \*Filepath = NULL;

struct stat sfileInfo;

if(argc != 3)

{

return -1;

}

Directory = (char \*)malloc(260);

Filename = (char \*)malloc(260);

Filepath = (char \*)malloc(260);

Directory = strcpy(Directory,argv[1]);

Filename = strcpy(Filename,argv[2]);

sprintf(Filepath,"%s/%s",Directory,Filename);

int result = access (Filepath, F\_OK);

if(result!=0)

{

iFd = creat(Filepath,0644);

FILE \*File;

char s[1000];

File=fopen(Filepath,"w");

while(1)

{

scanf("%s",s);

if(strcmp(s,"-1")==0) break;

fprintf(File,"%s\n",s);

}

fclose(File);

}

else

{

FILE \*File;

char s[1000];

File=fopen(Filepath,"w");

while(1)

{

scanf("%s",s);

if(strcmp(s,"-1")==0) break;

fprintf(File,"%s\n",s);

}

fclose(File);

}

close(iFd);

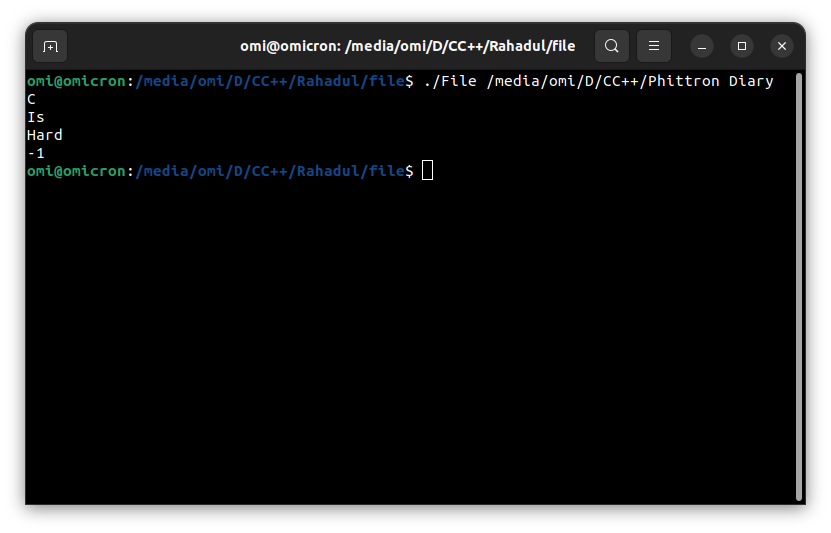
free(Directory);

free(Filename);

free(Filepath);

return(0);

}



TASK-2:

#include <stdio.h>

#include<unistd.h>

int main()

{

pid\_t child=fork();

if(child==0)

{

printf("I am child\n");

}

else

{

pid\_t grandchild=fork();

if(grandchild==0)

{

printf("I am parent\n");

}

else

{

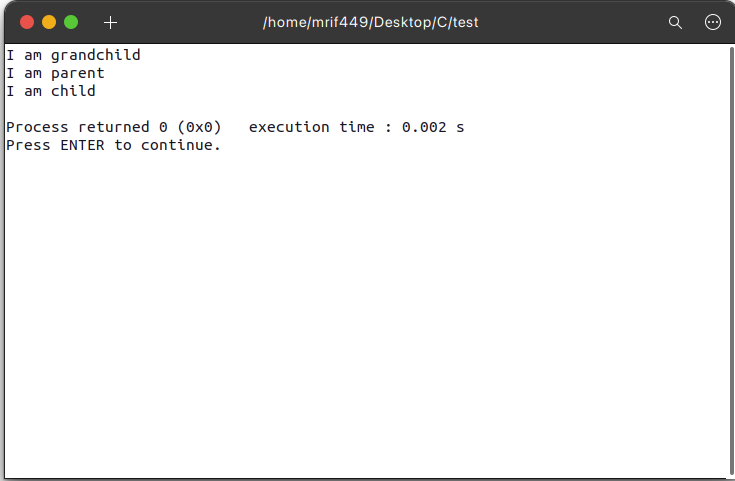
printf("I am grandchild\n");

}

}

return 0;

}



TASK-3:

#include <stdio.h>

#include<sys/types.h>

#include<unistd.h>

int main()

{

int a = fork();

int b = fork();

int c = fork();

int counter = 0;

if(getpid()%2!=0)

{

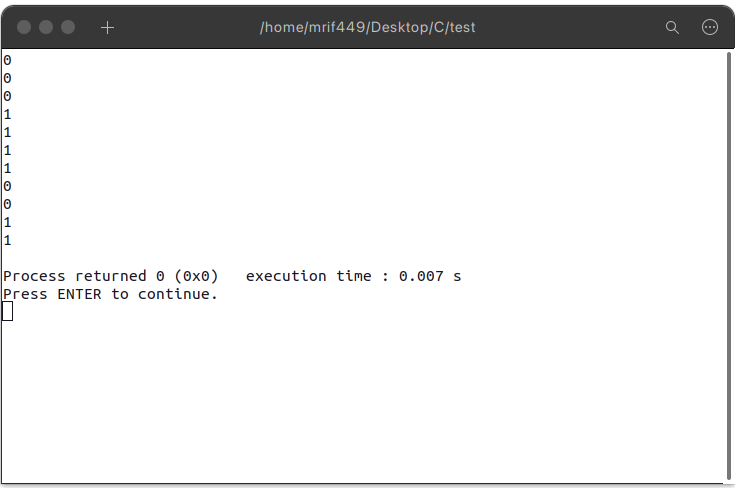
fork();

counter++;

}

printf("%d\n",counter);

}



TASK-4:

int main()

{

// array length assumed as 5

int temp\_var, x, y;

int len\_array = 5;

int array[len\_array];

for(int x=0; x<len\_array; x++)

{

scanf("%d",&array[x]);

}

temp\_var = 0;

len\_array = sizeof(array)/sizeof(array[0]);

for (x = 0; x<len\_array; x++)

{

for (y=x+1; y<len\_array;y++)

{

if (array[x]<array[y])

{

temp\_var=array[x];

array[x]=array[y];

array[y]=temp\_var;

}

}

}

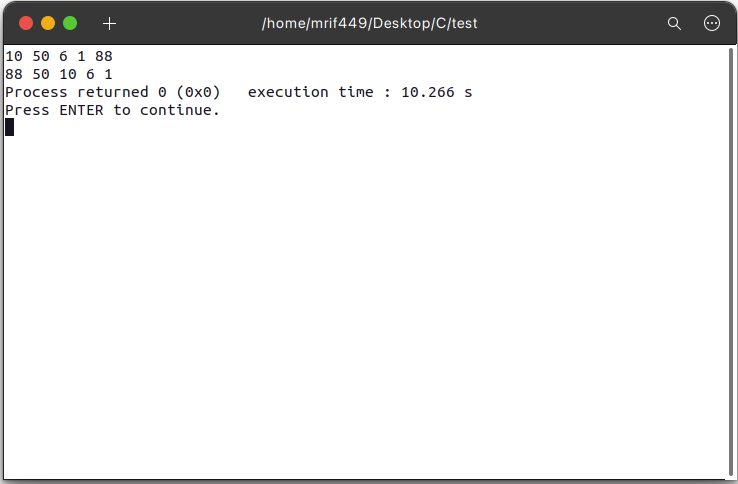
for (x=0; x<len\_array; x++)

{

printf("%d ",array[x]);

}

}



#include <stdio.h>

#include <string.h>

int main()

{

// array length assumed as 5

int temp\_var, x, y;

int len\_array = 5;

int array[len\_array];

for(int x=0; x<len\_array; x++)

{

scanf("%d",&array[x]);

}

temp\_var = 0;

len\_array = sizeof(array)/sizeof(array[0]);

for (x = 0; x<len\_array; x++)

{

if(array[x]%2 != 0)

{

printf("%d is ODD\n",array[x]);

}

else

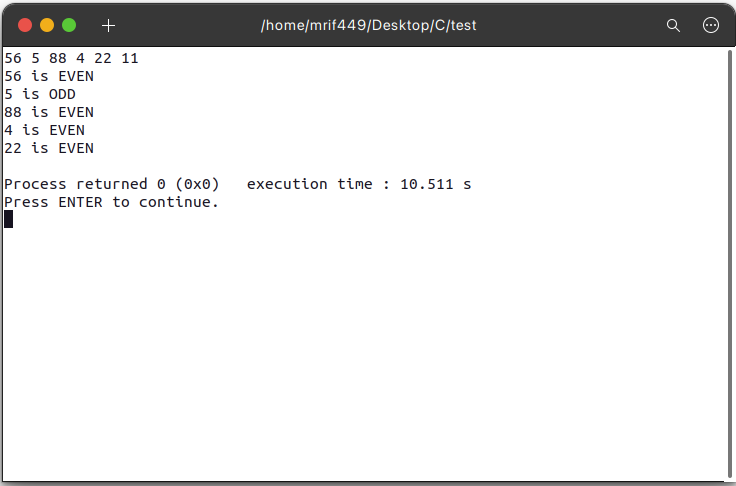
{

printf("%d is EVEN\n",array[x]);

}

}

}



TASK-5:

#include<stdio.h>

#include<stdlib.h>

#include<unistd.h>

#include<sys/types.h>

#include <stdio.h>

#include <unistd.h>

int main()

{

int pid, cpid, gpid;

int count = 0;

pid = fork();

if(pid == 0)

{

cpid = getpid();

pid = fork();

if(pid==0)

{

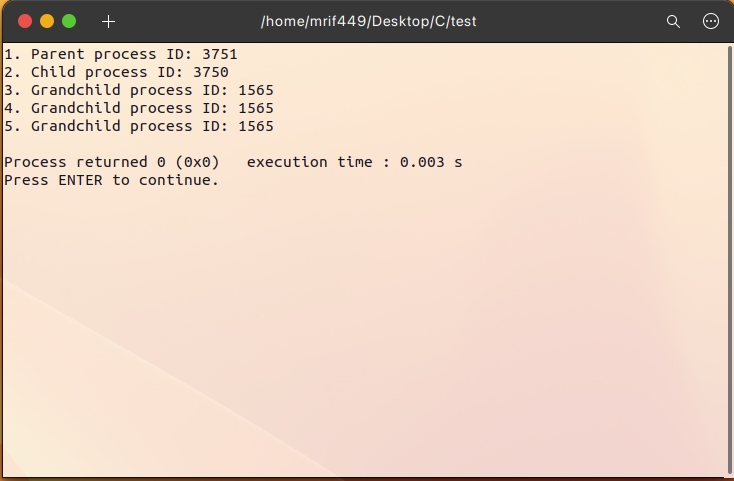
printf("1. Parent process ID: %d\n2. Child process ID: %d \n3. Grandchild process ID: %d\n4. Grandchild process ID: %d\n5. Grandchild process ID: %d\n", getpid(),cpid,getppid(),getppid(),getppid());

}

}

return 0;

}



//Threading

TASK-1:

#include <stdlib.h>

#include <unistd.h>

#include <pthread.h>

int thread\_counter = 1;

int thread\_number=1;

int sleep\_time = 1;

int runner = 5;

void \*threadFunction(void \*vargp )

{

sleep(sleep\_time);

printf("thread-%d running" , thread\_number);

}

int main()

{

for(int x=0; x < runner ; x++)

{

pthread\_t thread\_id;

pthread\_create(&thread\_id, NULL, threadFunction, NULL);

pthread\_join(thread\_id, NULL);

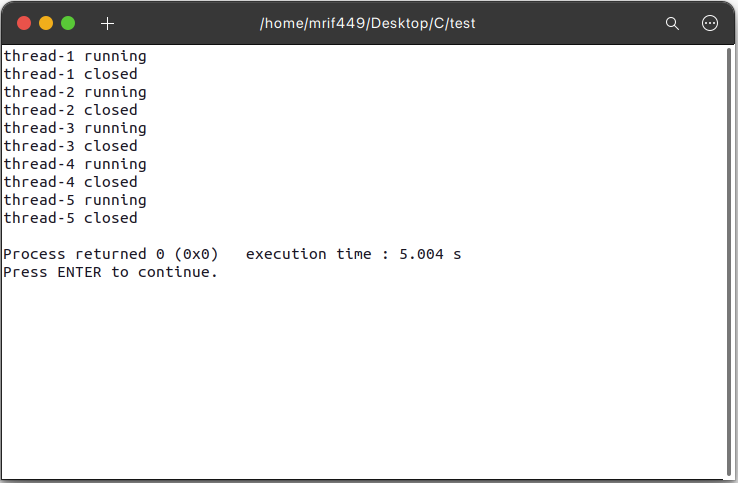
printf("\nthread-%d closed\n" , thread\_number);

thread\_number++ ;

}

return 0;

}



TASK-2:

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <pthread.h>

int counter=1;

void \*printer(void \*argv)

{

int \*thread\_number=(int\*)argv;

for(int i=0;i<5;++i)

{

printf("\nthread\_number %d prints %d\n",\*thread\_number,counter);

counter++;

}

}

int main()

{

pthread\_t t\_id1,t\_id2,t\_id3,t\_id4,t\_id5;

int t\_num1=1,t\_num2=2,t\_num3=3,t\_num4=4,t\_num5=5;

pthread\_create(&t\_id1,NULL, printer,(void\*)&t\_num1);

pthread\_create(&t\_id2,NULL, printer,(void\*)&t\_num2);

pthread\_create(&t\_id3,NULL, printer,(void\*)&t\_num3);

pthread\_create(&t\_id4,NULL, printer,(void\*)&t\_num4);

pthread\_create(&t\_id5,NULL, printer,(void\*)&t\_num5);

pthread\_join(t\_id1,NULL);

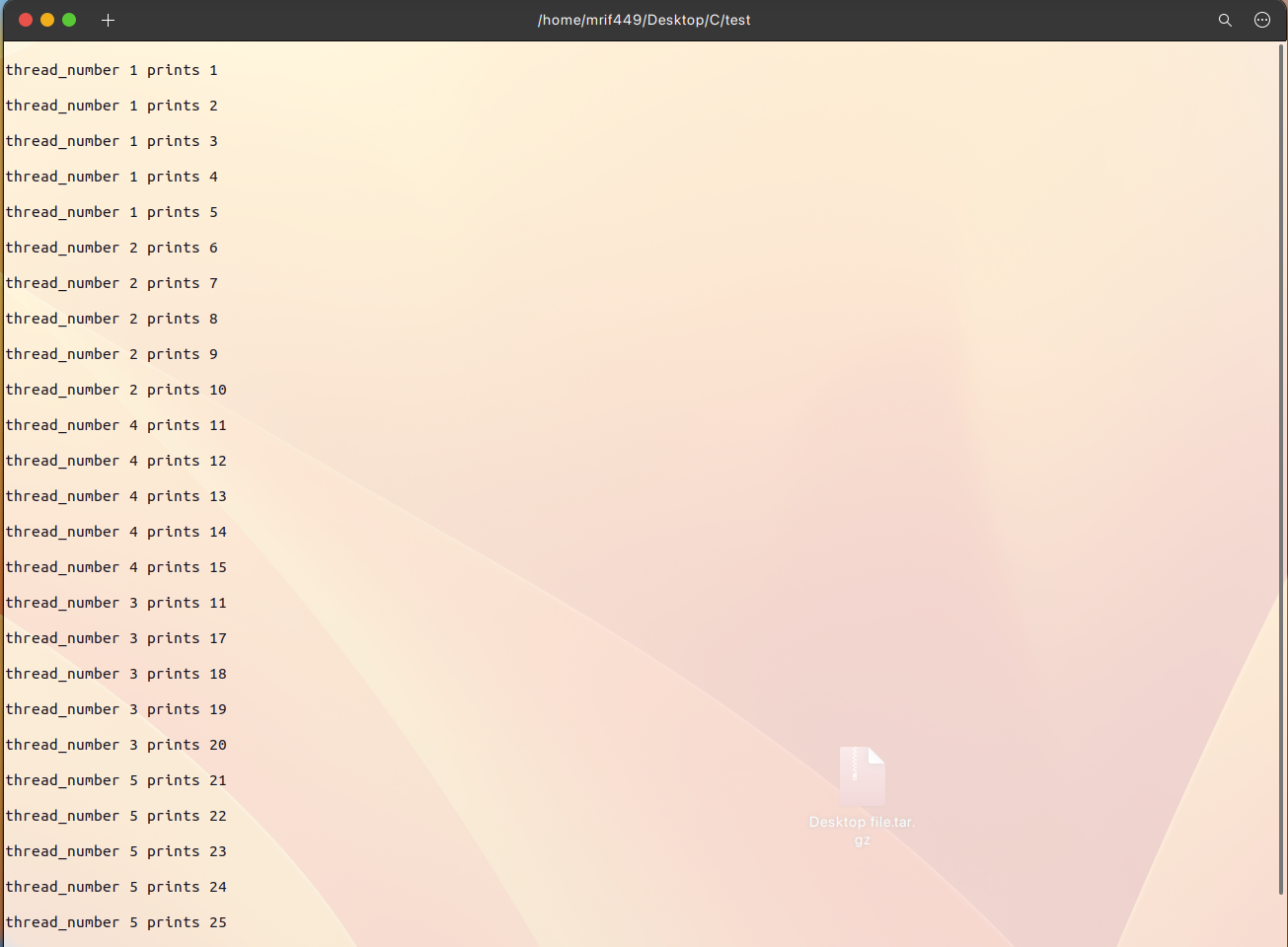
pthread\_join(t\_id2,NULL);

pthread\_join(t\_id3,NULL);

pthread\_join(t\_id4,NULL);

pthread\_join(t\_id5,NULL);

}



TASK-3:

#include <stdlib.h>

#include <stdio.h>

#include <pthread.h>

#include <string.h>

void \*ASCII\_counter(void \*input1)

{

char \*data = (char \*)input1;

int output = 0;

int \*final = malloc(sizeof(int));

int x = 0;

while (data[x] != '\0')

{

output += data[x];

x++;

}

\*final = output;

return (void \*)final;

}

int main(void)

{

char \*user\_name1="Rohim";

char \*user\_name2="Korim";

char \*user\_name3="Sumon";

pthread\_t value1, value2, value3;

pthread\_create(&value1, NULL, ASCII\_counter, user\_name1);

int \*number1;

pthread\_join(value1, (void \*\*)&number1);

pthread\_create(&value2, NULL, ASCII\_counter, user\_name2);

int \*number2;

pthread\_join(value2, (void \*\*)&number2);

pthread\_create(&value3, NULL, ASCII\_counter, user\_name3);

int \*number3;

pthread\_join(value3, (void \*\*)&number3);

if (\*number1 == \*number2 && \*number1 == \*number3)

{

printf("Youreka\n");

}

else if (\*number1 == \*number2 || \*number1 == \*number3 || \*number2 == \*number3)

{

printf("Miracle\n");

}

else

{

printf("Hasta la vista\n");

}

printf("%d\n", ASCII\_counter(user\_name1));

return 0;

}

