Week 4: Programs on pointers, arrays and multiple files

2021

Name:	SRN:	Section:
	Date:	Week Number:

MAKEFILE USED:

```
#variable declaration
CC=gcc
CFLAGS=-c -Wall
#basically variables are referenced anywhere in the file
#using $(VARIABLE NAME)
basefile=bruh
secondfile=monke


#target:dependancy
#	command


1:$(basefile)1.o $(secondfile)1.o #write all of ur dependancies
	$(CC) $(basefile)1.o $(secondfile)1.o -o $(basefile)1

$(basefile)1.o:$(basefile)1.c
	$(CC) $(CFLAGS) $(basefile)1.c

$(secondfile)1.o:$(secondfile)1.c
	$(CC) $(CFLAGS) $(secondfile)1.c

clean:
	del (basefile)1 .exe



2:$(basefile)2.o $(secondfile)2.o #write all of ur dependancies
	$(CC) $(basefile)2.o $(secondfile)2.o -o $(basefile)2

$(basefile)2.o:$(basefile)2.c
	$(CC) $(CFLAGS) $(basefile)2.c
```

```
$(secondfile)2.o:$(secondfile)2.c
	$(CC) $(CFLAGS) $(secondfile)2.c

clean:
	del (basefile)2 .exe



3:$(basefile)3.o $(secondfile)3.o #write all of ur dependancies
	$(CC) $(basefile)3.o $(secondfile)3.o -o $(basefile)3

$(basefile)3.o:$(basefile)3.c
	$(CC) $(CFLAGS) $(basefile)3.c

$(secondfile)3.o:$(secondfile)3.c
	$(CC) $(CFLAGS) $(secondfile)3.c

clean:
	del (basefile)3 .exe



4:$(basefile)4.o $(secondfile)4.o #write all of ur dependancies
	$(CC) $(basefile)4.o $(secondfile)4.o -o $(basefile)4


$(basefile)4.o:$(basefile)4.c
	$(CC) $(CFLAGS) $(basefile)4.c

$(secondfile)4.o:$(secondfile)4.c
	$(CC) $(CFLAGS) $(secondfile)4.c

clean:
	del (basefile)4 .exe



5:$(basefile)5.o $(secondfile)5.o #write all of ur dependancies
	$(CC) $(basefile)5.o $(secondfile)5.o -o $(basefile)5

$(basefile)5.o:$(basefile)5.c
	$(CC) $(CFLAGS) $(basefile)5.c

$(secondfile)5.o:$(secondfile)5.c
```

```
	$(CC) $(CFLAGS) $(secondfile)5.c

clean:
	del (basefile)5.exe
```

1	Write a function to display an array elements in the reverse order using multiple files. a) using index b) using pointer **Input:** Enter the size of an array 5 Enter elements 11 22 33 44 55 **Output:** Array elements: 11 22 33 44 55 Reversed array: 55 44 33 22 11

	Program:
	bruh1.c
	#include<stdio.h>
	
	void revarr(int *arr,int arrlenght);
	void revpointer(int *arr,int arrlenght);
	
	int main(){
	int arrlenght;
	printf("Enter lenght of array :");
	scanf("%d",&arrlenght);
	printf("\n\n");
	
	int arr[arrlenght];
	for (int i = 0; i < arrlenght; i++)
	{
	printf("Enter element:");

```
        scanf("%d",&arr[i]);
    }

    revarr(arr,arrlenght);
    revpointer(arr,arrlenght);

    return 0;
}
```

monke1.c

```
#include<stdio.h>

void revarr(int *arr,int arrlenght);
void revpointer(int *arr,int arrlenght);

int main(){
    int arrlenght;
    printf("Enter lenght of array :");
    scanf("%d",&arrlenght);
    printf("\n\n");

    int arr[arrlenght];
    for (int i = 0; i < arrlenght; i++)
    {
        printf("Enter element:");
        scanf("%d",&arr[i]);
    }

    revarr(arr,arrlenght);
    revpointer(arr,arrlenght);

    return 0;
}
```

Output Screenshot:

2	Write a function for factorial using recursion and use it to find C(n, r) using multiple files. **Input:** 5 2 **Output:** ncr is: 10

	Program:
	bruh2.c
	#include<stdio.h>
	int fact(int n);
	
	int main(){
	int n,r;
	printf("Enter n and r:");
	scanf("%d %d",&n,&r);
	
	int ans=fact(n)/(fact(n-r)*fact(r));
	printf("nCr = %d",ans);
	return 0;
	
	}
	monke2.c
	#include<stdio.h>
	
	int fact(int n){
	if (n<=1) return 1;
	else{
	return n *fact(n-1);
	}
	}

Output Screenshot:

3 Write a C program to print all unique elements of an array using Make file

Input:

Input the number of elements to be stored in the array: 5
Input 5 elements in the array :
element - 0 : 1
element - 1 : 2
element - 2 : 1
element - 3 : 3
element - 4 : 3

Output:

The unique elements found in the array are:
List of Unique Array Elements in this Array are : 2

Program:

bruh3.c

```
#include<stdio.h>

int uniquenum(int *arr,int arrlen);
int unique(int n,int *arr,int arrlen);

int main(){
   int arrlen;
   printf("Enter length of array :");
   scanf("%d",&arrlen);
   printf("\n\n");

   int arr[arrlen];
   for (int i = 0; i < arrlen; i++)
   {
      printf("Enter element:");
      scanf("%d",&arr[i]);
   }

   printf("the unique elements found in the array are:");
   printf("No of unique elements: %d",uniquenum(arr,arrlen));
   return 0;



}
```

monke3.c

```
#include<stdio.h>

int unique(int n,int *arr,int arrlen){
    int count=0;
    for (int i = 0; i < arrlen; i++)
    {
        if(n==arr[i]){
            count++;
            if (count>1)
            {
                return 0;
            }
        }
    }
    return 1;
}

int uniquenum(int *arr,int arrlen){
    int count=0;
    for (int i = 0; i < arrlen; i++)
    {
        if(unique(arr[i],arr,arrlen)==1){
            printf(" %d ",arr[i]);
            count++;
        }
    }
    printf("\n");
    return count;

}
```

Output Screenshot:

4

Write a C program to Calculate the power of any number using recursion and multiple files

Input:

Recursion : Calculate the power of any number :

Input the base value : 4

Input the value of power : 2

Output:

The value of 4 to the power of 2 is : 16

Program:

bruh4.c

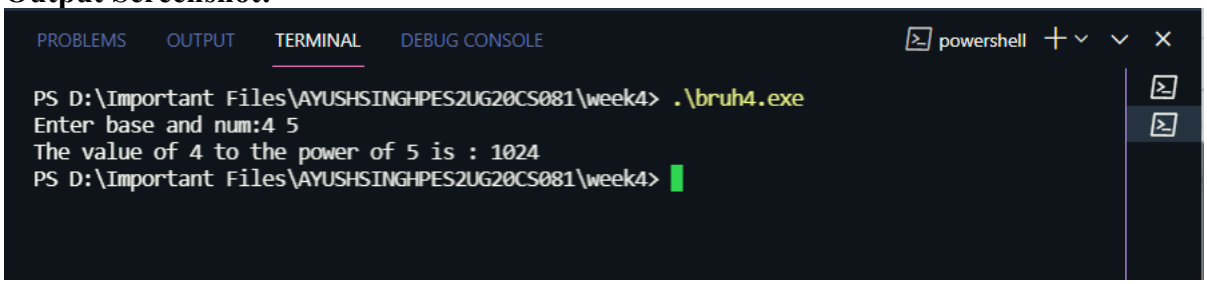
```
#include<stdio.h>

int powe(int base,int num);

int main(){
    int base ,num;
    printf("Enter base and num:");
    scanf("%d %d",&base,&num);

    printf("The value of %d to the power of %d is : %d",base,num,powe(base,num));
    return 0;

}
```

monke4.c

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

```
int powe(int base,int num){
    if(num<=1) return base;
    else return base*powe(base,num-1);
}
```

Output Screenshot:



5 Write a function to check whether a given number is prime and use that to find the next prime number, greater than a given number.

Input1:
Enter a number
4
Output1:
Next prime number=5
Input2:
Enter a number
113
Ouput2:
Next prime number=127

Program:

bruh5.c

```
#include<stdio.h>

int checkprime(int n);
int nextprime(int n);

int main(){
    int n;
    printf("Enter Number:");
    scanf("%d",&n);
    printf("\n\n");
```

```
    printf("IFPRIME: %d\n",checkprime(n));
    printf("NEXT PRIME: %d",nextprime(n));

    return 0;
}
```

monke5.c

```
#include <stdio.h>

int checkprime(int n)
{
    if (n > 1)
    {
        for (int i = 2; i < n / 2; i++)
        {
            if (n % i == 0)
                return 0;
        }
        return 1;
    }
    else
        return 0;
}

int nextprime(int n)
{
    int flag = 1;
    while (flag)
    {
        n++;
        if (checkprime(n) == 1)
        {
            flag = 0;
            return n;
        }

    }
}
```

Output Screenshot: