Lab assignment-4

Note:- In all subsequent codes, assume that:

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

header files have been included.

① a) Run the process where child process prints "Child" and parent prints "Parent", with their ids.

```
Void main()
{
  pid_t pid = fork();
  if (pid == 0)
  {
    printf("Child: %d\n", getpid());
    while(1);
  }
  else
  {
    printf("Parent: %d\n", getpid());
    while(1);
  }
}
```

b) Terminate the child process.

```
void main(){
  pid_t pid = fork();
  if(pid == 0){
    printf("Child: %d\n", getpid());
    _exit(0);
  }
  else{
    printf("Parent: %d\n", getpid());
    while(1);
  }
}
```

c) Terminate the parent process.

```
void main() {
    pid_t pid = fork();
    if (pid == 0){
        printf("Child: %d\n", getpid());
        while (1);
    }
    else {
        printf("Parent: %d\n", getpid());
        exit(0);
    }
    printf("Pid: %d\n", getpid());
}
```

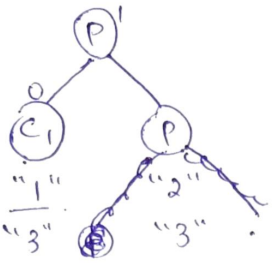
d) Make parent process wait for child to complete its task.

```
void main() {
    pid_t pid = fork();
    if (pid == 0){ printf("Child: %d\n", getpid()); }
    else {
        printf("Parent: %d\n", getpid());
        wait(NULL);
    }
    printf("Pid: %d\n", getpid());
}
```

e) Terminate child process.

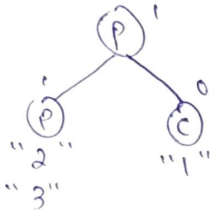
```
void main() {
    pid_t pid = fork();
    if (pid == 0) {
        printf("Child: %d\n", getpid());
        exit(0);
    }
    else{
        printf("Parent: %d\n", getpid());
        wait(NULL);
    }
    printf("Pid: %d\n", getpid());
}
```

(2) a)



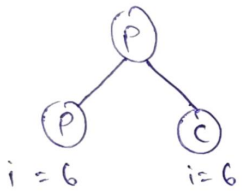
O/P: 2313

b)



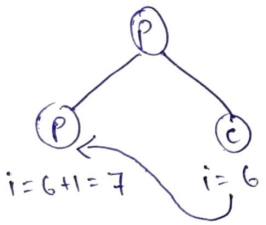
O/P = 123

c) i = 5



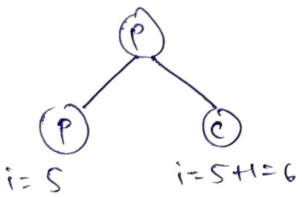
O/P = Child: 6 Parent: 6

d) i = 5



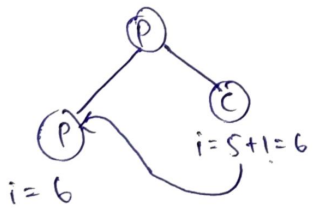
O/P: Child: 6 Parent: 7

e) i = 5



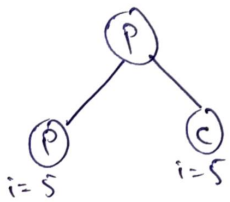
O/P: Child: 6 Parent: 5

f) i = 5



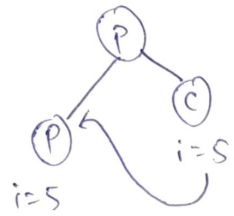
O/P: Child: 6 Parent: 6

g) i = 5



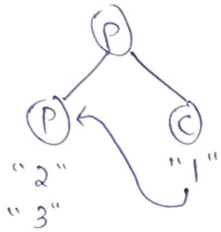
O/P: Child: 5 Parent: 5

h) i=5



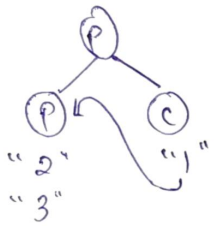
O/P: Child: 5 Parent: 5

i)



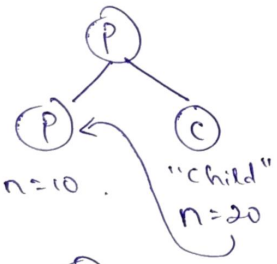
O/P: 123

j)



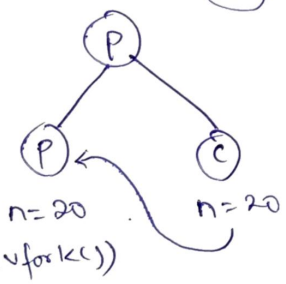
O/P: 123

k)



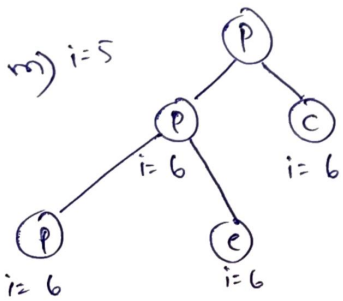
O/P: child
n=20
Parent
n=10

l)



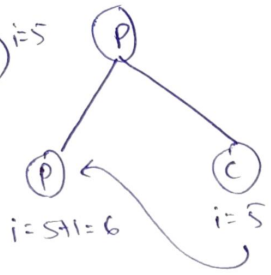
O/P: Child
n=20
Parent
n=20

m) i=5



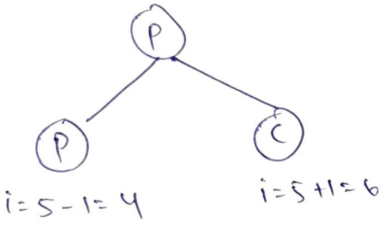
O/P: 6 6 6 6

n) i=5



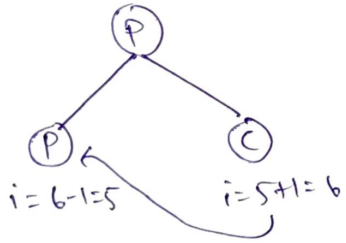
o/p: Child: 5 Parent: 6

o) i=5



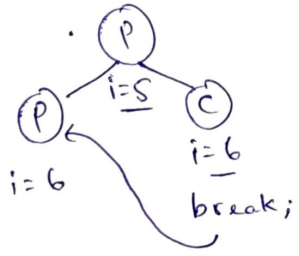
o/p: 46

p) i=5



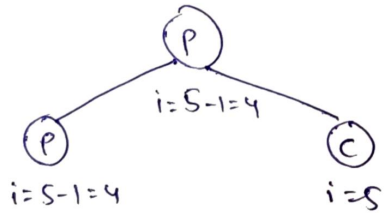
o/p: 5

q) j, i=5



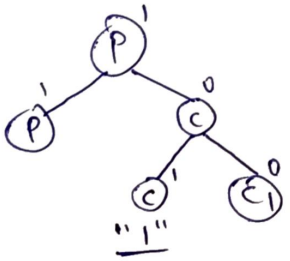
o/p: 665

r) j, i=5



o/p: 454

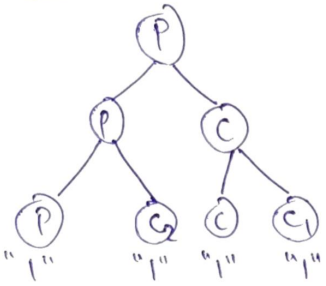
s)



o/p: 1

t) Next page

d) fun1()



∴ 4 processes will be returned to main().

main()

O/P

1
1
1
1
1
1
1
1

③ Wap to create 3 child processes to :
- copy contents of file1 to file2
- display contents of file2
- display sorted content of file2 in reverse order.

```
int main()
{
    if (fork() > 0) {
        sleep(1);
    }
    else {
        printf("Child1 id: %d Parent id: %d \n", getpid(), getppid());
        printf("\n\n Copy the contents of f1.txt to f2.txt\n");
        execlp("cp","cp","-f","f1.txt","f2.txt",NULL);
    }
    if(fork()>0)
        sleep(1);
    else {
        printf("Child 2 id: %d Parent id: %d \n", getpid(), getppid());
        printf("\n\nDisplay the contents of f2.txt\n");
        execlp("cat","cat","f2.txt",NULL);
    }
    if(fork()>0)
        sleep(1);
    else {
        printf("Child 3 id: %d Parent id: %d \n", getpid(), getppid());
        printf("\n\nSort the contents of f2.txt\n");
        execlp("sort","sort","-r","f2.txt",NULL);
    }
    return 0;
}
```

(4) Wap to generate a Fibonacci series and store it in an array; and then display prime numbers in the series.

```
int main() {
    int sz;
    printf("Enter the length of the fibonacci series:");
    scanf("%d", &sz);
    int arr[sz];
    if (vfork() == 0) {
        int m1 = 0, m2 = 1;
        for (int i=0; i<sz; i++)
        {
            arr[i] = m1;
            m2 = m1 + m2;
            m1 = m2 - m1;
        }
        exit(0);
    }
    else {
        wait(NULL);
        printf("Array of %d length of fibonacci series is:", sz);
        for (int i=0; i<sz; i++)
            printf("%d ", arr[i]);
        printf("\n");
        int m, count;
        printf("Prime numbers at different locations:\n");
        for(int i=1; i<sz; i++){
            count = 1;
            for (m=2; m<arr[i]; m++) {
                if (arr[i] % m == 0) count++;
            if (count == 1)
                printf("%d is a prime fibonacci number at %d\n", arr[i], i+1);
    }
    return 0;
}
```