P.S. Assignment - 1

Q1) Write a c program to create a child process using fork() system call. The child process will print the message "child" with its process identifier and then continue in an indefinite loop. The parent process will print the message "Parent" with its process identifier and then continue in an indefinite loop.

a) Run the program and trace the state of both process.

b) Terminate the child process. Then trace the state of processors.

c) Run the program and trace the state of both processes. Terminate the parent process. Then trace the state of processes.

d) Modify the program so that the parent process after displaying the message will wait for child process to complete its task. Again run the program and trace the state of both processes.

e) Terminate the child process. Then trace the state of processes.

Program :-

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

int main() {

    if (fork() == 0) {

        printf("child %d \n", getpid());

        while (1);

    }

    else {

        printf("Parent %d \n", getpid());
        wait(NULL);
        while (1); }
```

Scanned with CamScanner

```
    return 0;
}
void a() {
    if (fork() == 0) {
        printf ("child : %d \n", getpid());
    }
    else
        printf ("Parent: %d \n", getpid());
}
void b() {
    if (fork() == 0) {
        printf ("child : %d \n", getpid());
        exit(0);
    }
    else
        printf ("Parent : %d \n", getpid());
}
void c() {
    if (fork() == 0) {
        printf ("child : %d \n", getpid());
    else {
        printf ("Parent: %d \n", getpid());
        exit(0);
    }
}
void d() {
    if (fork() == 0) {
        printf ("child: %d \n", getpid());
    }
    else {
```

Scanned with CamScanner

```
        printf("Parent: %d \n", getpid());
        wait(NULL);
    }
}
void c(){
    if(fork() == 0) {
        printf("child: %d \n", getpid());
        exit(0);
    }
    else {
        printf("Parent: %d \n", getpid());
        wait(NULL);
    }
}
int main(){
    a();
    b();
    c();
    d();
    e();
    return 0;
}
```

Output :-

a) Parent : 29171
   child : 29172

b) Parent : 29218
   child: 29219

Scanned with CamScanner

c) Parent : 29491

child : 29492

d) Parent : 29640

child : 29641

e) Parent : 29772

child : 29773

Q2) Trace the output of the following program segment :

a)
```
int main()
{
  if (fork() == 0)
    fprintf("1");
  else
    fprintf("2");
  fprintf("3");
  return 0;
}
```

output :- 2313

b)
```
int main()
{
  if (fork() == 0)
    fprintf("1");
  else
    fprintf("2");
  fprintf("3");
}
```

output :- 1323

timeout : the monitored command dumped core.

Scanned with CamScanner

c) int main()

```
{
  pid_t pid;
  int i = 5;
  pid = fork();
  i = i + 1;
  if (pid == 0)
  {
    printf ("child : %d", i);
  }
  else
  {
    wait (NULL);
    printf ("Parent : %d", i);
  }
  return 0;
}
```

output :- child : 6
Parent : 6

d) int main()

```
{
  pid_t pid;
  int i = 5;
  pid = vfork ();
  i = i + 1;
  if (pid == 0)
  {
    printf ("child : %d", i);
    exit(0);
  }
  else
  {
    printf ("Parent : %d", i);
  }
  return 0;
}
```

Scanned with CamScanner

output :- child : 6
Parent : 7

e)
```
int main()
{
  pid_t pid;
  int i = 5;
  pid = fork();
  if (pid == 0)
  {
    i = i + 1;
    printf ("child : %d", i);
  }
  else
  {
    wait (NULL);
    printf ("Parent : %d", i);
  }
  return 0;
}
```

output :- child : 6
Parent : 5

f)
```
int main()
{
  pid_t pid;
  int i = 5;
  pid = vfork();
  if (pid == 0)
  {
    i = i + 1;
    printf ("child : %d", i);
    exit(0);
  }
```

Scanned with CamScanner

```
    else {
      printf("Parent: %d", i);
    }
    return 0;
}
```

output :- child : 6
Parent : 6

g) int main()

```
{
  int i = 5;
  if(fork() == 0)
  {
    printf("child: %d", i);
  }
  else
  {
    printf("Parent: %d", i);
  }
  return 0;
}
```

output :- Parent : 5
child : 5

h) int main()

```
{
  int i = 5;
  if(vfork() == 0)
  {
    printf("child: %d", i);
    exit(0);
  }
  else
  {
    printf("Parent: %d", i);
  }
  return 0;
}
```

output :- child : 5
Parent : 5

Scanned with CamScanner

i) int main()

```
{
  if (fork() == 0)
  {
    printf("1");
  }
  else
  {
    wait(NULL);
    printf("2");
    printf("3");
  }
  return 0;
}
```

output :- 1 2 3

j) int main()

```
{
  if (vfork() == 0)
  {
    printf("1");
    exit(0);
  }
  else
  {
    printf("2");
    printf("3");
  }
  return 0;
}
```

output :- 1 2 3

k) int main()

```
{
  pid_t c1;
  int n = 10;
  c1 = fork();
```

Scanned with CamScanner

```
if (c1 == 0)
{
  printf(" child \n");
   n = 20
   printf (" n = %d \n", n);
}
else
{
   wait (NULL);
   printf (" Parent \n");
   printf(" n = %d \n ",n);
}
return 0;
}
```

output :- child
n = 20
Parent
n = 10

l) int main()

```
{
  pid_t c1;
  int n = 10;
  c1 = vfork();
  if (c1 == 0)
  {
    printf (" child \n");
    n = 20;
    printf (" n = %d \n", n);
    exit(0);
  }
  else
  {
    printf (" Parent \n");
    printf (" n = %d \n", n);
  }
  return 0;
}
```

output :- child
n = 20
Parent
n = 20

Scanned with CamScanner

m) int main()

```
{ int i = 5;
  fork();
  i = i + 1;
  fork();
  fprintf(stderr, "%d", i);
  return 0;
}
```

output :- 6 6 6 6

n) int main()

```
{
  pid_t pid;
  int i = 5;
  pid = vfork();
  if (pid == 0)
  {
    printf("child: %d", i);
    exit(0);
  }
  else
  {
    i = i + 1;
    printf("Parent: %d", i);
  }
  return 0;
}
```

output :- child : 5
Parent : 6

o) int main()

```
{
  int i = 5;
  if (fork() == 0)
    i = i + 1;
```

Scanned with CamScanner

```
else
    i = i - 1;
    fprintf (stderr, "%d", i);
    return 0;
}
```

output :- 46

b)
```
int main()
{
    int i = 5;
    if (vfork() == 0)
    {
        i = i + 1;
        _exit (0);
    }
    else
        i = i + 1;
    fprintf (stderr, "%d", i);
    return 0;
}
```

output :- 5

c)
```
int main()
{
    int j, i = 5;
    for (j = 1; j < 3; j++)
    {
        if (fork() == 0)
        {
            i = i + 1;
            break;
        }
        else
            wait (NULL);
    } fprintf (stderr, "%d", i); return 0;
}
```

output :- 6 6 5

Scanned with CamScanner

g) int main()
{
int j, i = 5
for (j = 1; j < 3; j++)
{
if (fork() != 0)
{ i = i + 1;
break;
}
}
fprintf (stderr, "%d", i);
return 0;
}

output :- 4 4 5

h) int main()
{
if (fork() == 0)
if (fork())
printf ("a \n");
return 0;
}

output :- 1

i) void fun1 () {
fork();
fork();
printf ("a \n");
}
int main() {
fun1 ();
printf ("a \n");
return 0;
}

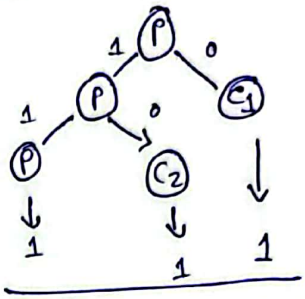
output :-
1
1
1
1
1
1
1
1
11

Scanned with CamScanner

Q3) Trace the following program segment & determine how many processes are created. Draw a graph that shows how the processes are related.

a)
```
int main()
{
    if(fork() && fork());
    printf("1");
    return 0;
}
```

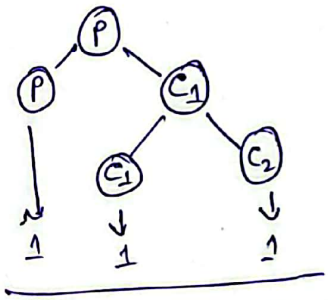
output :- 1 1 1



Print (111) => O/P = 111

b)
```
int main()
{
    if (fork() || fork());
    printf("1");
    return 0;
}
```

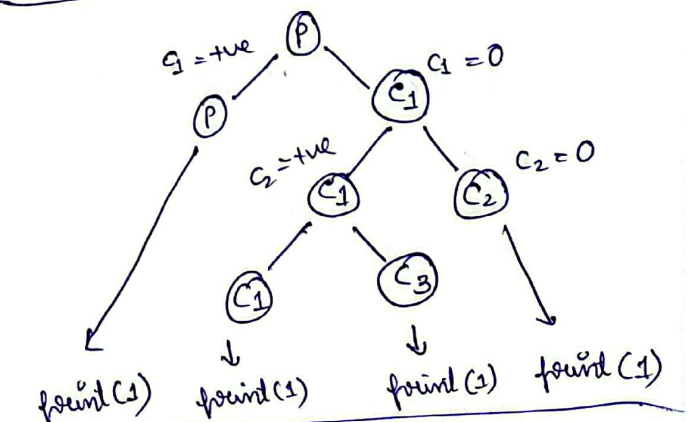
output :- 1 1 1



Print (111) => O/P = 111

c)
```
int main()
{
    pid_t c1, c2;
    c2 = 0;
    c1 = fork();
    if (c1 == 0)
        c2 = fork();
    if (c2 > 0)
        fork();
    printf("1");
    return 0;
}
```

output :- 1 1 1 1

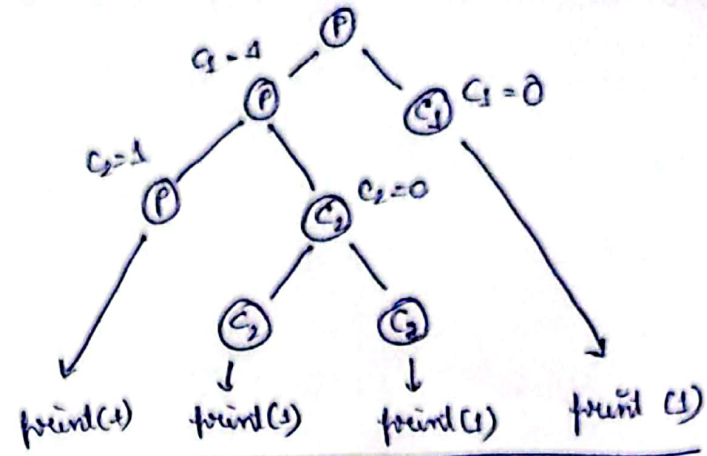


O/P = 1 1 1 1

Scanned with CamScanner

d) int main()
{
  pid_t c1=1, c2=1;
  c1 = fork();
  if (c1 != 0)
    c2 = fork();
  if (c2 == 0)
    fork();
  printf("1");
  return 0;
}

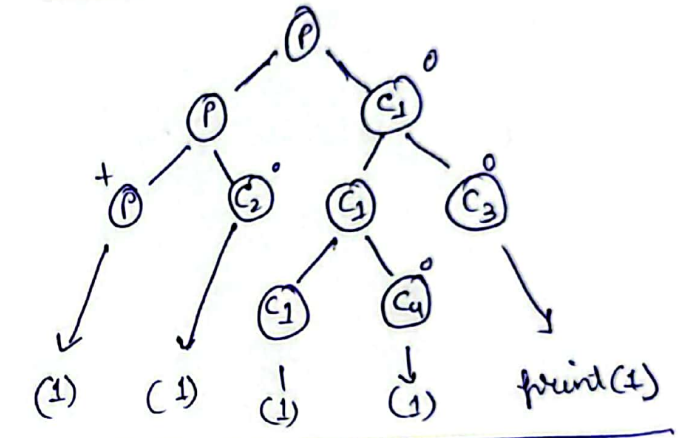
output :- 1 1 1 1



O/P = 1 1 1 1

e) int main()
{
  if (fork() || fork())
    fork();
  printf("1");
  return 0;
}

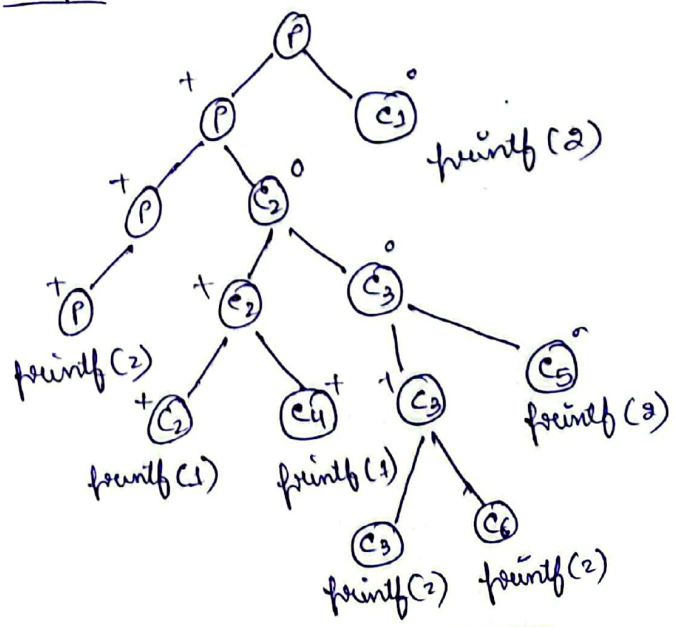
output :- 1 1 1 1 1



O/P = 1 1 1 1 1

f) int main()
{
  if (fork() && (!fork()))
  {
    if (fork() || fork())
    {
      fork();
    }
  }
  printf("2");
  return 0;
}

output :- 2 2 2 2 2 2 2



O/P = 2 2 2 2 2 2 2

Scanned with CamScanner