

NAME: SPANDAN MUKHERJEE

REGISTRATION NUMBER: 21BCE1132

SUBJECT: OPERATING SYSTEM LAB – 2

- 1) Develop a child process to display the hello message, observe the output and give justification of the output.

The screenshot shows a code editor window titled 'hello.c' with the following C code:

```
1 // Develop a child process to display the hello message, observe the output and give
2 justification of the output.
3 #include <stdio.h>
4 #include <sys/types.h>
5 #include <unistd.h>
6
7 int main()
8 {
9     fork();
10    printf("Hello World!\n");
11    return 0;
12 }
```

Overlaid on the code editor is a terminal window titled 'spandan@spandan-VirtualBox: ~'. The terminal shows the following commands and output:

```
spandan@spandan-VirtualBox:~$ gedit hello.c
^C
spandan@spandan-VirtualBox:~$ gcc hello.c
spandan@spandan-VirtualBox:~$ ./a.out
Hello World!
Hello World!
spandan@spandan-VirtualBox:~$
spandan@spandan-VirtualBox:~$
spandan@spandan-VirtualBox:~$ gedit hello.c
```

- 2) Develop a child process to display the numbers 500 to 1000 and parent process to display from 1 to 500. observe the output and give justification for the same.

The screenshot shows a code editor window titled 'demo2.c' with the following C code:

```
1 #include <stdio.h>
2 #include <unistd.h>
3 #include <sys/types.h>
4
5 int main(){
6     pid_t k;
7     k = fork();
8     if(k==0){
9         for(int i=500;i<1001;i++){
10             printf("%d ",i);
11         }
12     }
13     else if(k>0){
14         for(int i=1;i<501;i++){
15             printf("%d ",i);
16         }
17     }
18     return 0;
19 }
```

```
spandan@spandan-VirtualBox:~$ gedit demo2.c
^C
spandan@spandan-VirtualBox:~$ gcc demo2.c
spandan@spandan-VirtualBox:~$ ./a.out
500 501 502 503 504 505 506 507 508 509 510 511 512 513 514 515 516 517 518 519 520 521 522 523 524 525 526 527 528 5
29 530 531 532 533 534 535 536 537 538 539 540 541 542 543 544 545 546 547 548 549 550 551 552 553 554 555 556 557 55
8 559 560 561 562 563 564 565 566 567 568 569 570 571 572 573 574 575 576 577 578 579 580 581 582 583 584 585 586 587
588 589 590 591 592 593 594 595 596 597 598 599 600 601 602 603 604 605 606 607 608 609 610 611 612 613 614 615 616
617 618 619 620 621 622 623 624 625 626 627 628 629 630 631 632 633 634 635 636 637 638 639 640 641 642 643 644 645 6
46 647 648 649 650 651 652 653 654 655 656 657 658 659 660 661 662 663 664 665 666 667 668 669 670 671 672 673 674 67
5 676 677 678 679 680 681 682 683 684 685 686 687 688 689 690 691 692 693 694 695 696 697 698 699 700 701 702 703 704
705 706 707 708 709 710 711 712 713 714 715 716 717 718 719 720 721 722 723 724 725 726 727 728 729 730 731 732 733
734 735 736 737 738 739 740 741 742 743 744 745 746 747 748 749 750 751 752 753 754 755 1 2 3 4 5 6 7 8 9 10 11 12 13
14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52
53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91
92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122
123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 1
52 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 18
1 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 209 210
211 212 213 214 215 216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239
240 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 2
69 270 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 288 289 290 291 292 293 294 295 296 297 298
299 300 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323 324 325 326 327
328 329 330 331 332 333 334 335 336 337 338 339 340 341 342 343 344 345
spandan@spandan-VirtualBox:~$ spandan@spandan-VirtualBox:~$ spandan@spandan-VirtualBox:~$ spandan@spandan-VirtualBox:~$
spandan@spandan-VirtualBox:~$
```

- 3) Show the code to create the Zombie and orphan processes with the explanation of how zombie process differs from orphan process.

```
Terminal
demo3.c
1 // Zombie Process code
2
3 #include<stdio.h>
4 #include<unistd.h>
5 #include<sys/wait.h>
6
7 int main()
8 {
9     int pid = fork();
10
11     if(pid==0)
12     {
13         printf("child process id: %d has parent id : %d \n",getpid() ,getppid());
14     }
15     else if(pid>0)
16     {
17         sleep(50);
18         printf("parent process id: %d has grand parent id : %d \n",getpid() ,getppid());
19     }
20     else
21     {
22         printf("process not created");
23     }
24
25     return 0;
26 }
27

spandan@spandan-VirtualBox:~$ gedit demo3.c
^C
spandan@spandan-VirtualBox:~$ gcc demo3.c
spandan@spandan-VirtualBox:~$ ./a.out
child process id: 4810 has parent id : 4809
^C
spandan@spandan-VirtualBox:~$ gedit demo3.c
```

The screenshot shows a Linux desktop environment. A code editor window titled 'demo4.c' contains the following C code:

```
1 // Orphan process code
2
3 #include <stdio.h>
4 #include <stdlib.h>
5 #include <sys/types.h>
6 #include <unistd.h>
7
8 int main()
9 {
10     int pld;
11     pld = fork();
12
13     if(pld == 0)
14     {
15         printf("I am the child, my process ID is %d\n", getpid());
16         printf("My parent's process ID is %d\n", getppid());
17         sleep(30);
18         printf("\nAfter sleep\nI am the child, my process ID is %d\n", getpid());
19         printf("My parent's process ID is %d\n", getppid());
20         exit(0);
21     }
22     else
23     {
24         sleep(20);
25         printf("I am the parent, my process ID is %d\n", getpid());
26         printf("The parent's parent, process ID is %d\n", getppid());
27         printf("Parent terminates\n");
28     }
29     return 0;
30 }
```

A terminal window titled 'spandan@spandan-VirtualBox' shows the execution of the program:

```
spandan@spandan-VirtualBox:~$ gcc demo4.c
spandan@spandan-VirtualBox:~$ ./a.out
I am the child, my process ID is 5211
My parent's process ID is 5212
After sleep
I am the child, my process ID is 5211
My parent's process ID is 5212
I am the parent, my process ID is 5211
The parent's parent, process ID is 5211
Parent terminates
```

- 4) Develop a child process and parent process to count the vowels and consonants accordingly of the given string and display the output.

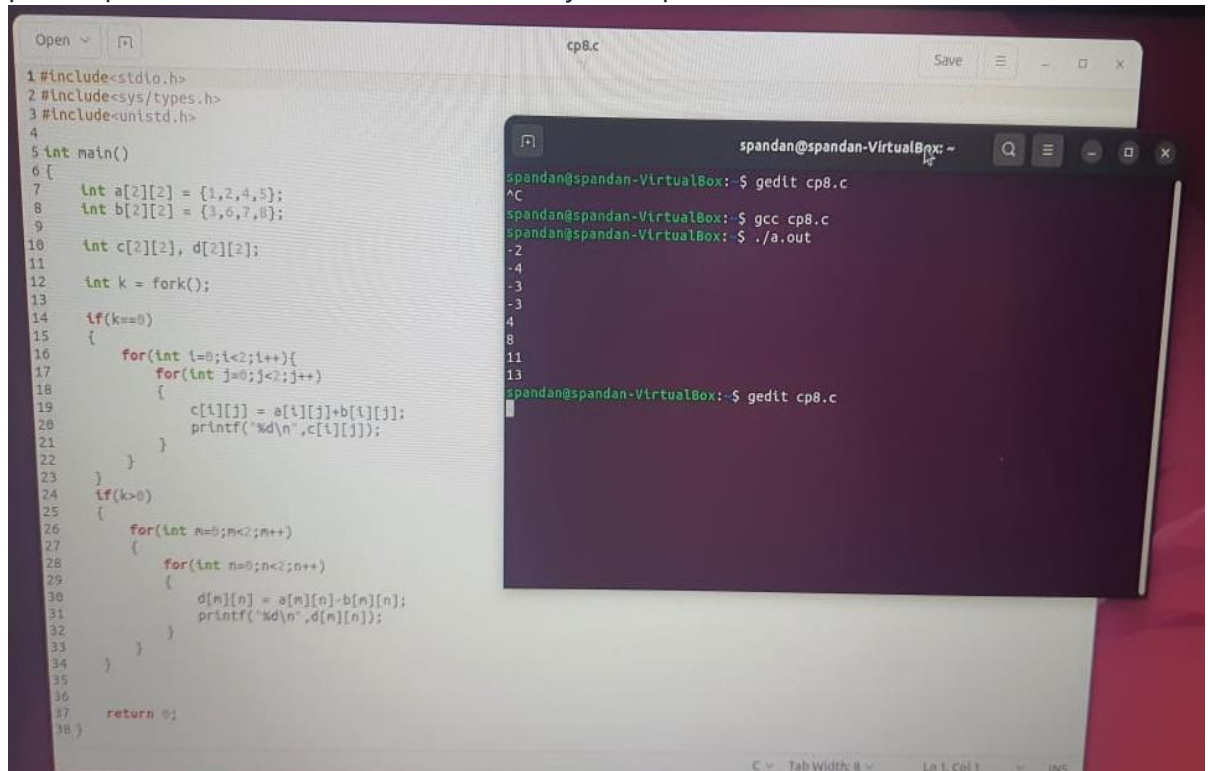
The screenshot shows a Linux desktop environment. A code editor window titled 'vow.c' contains the following C code:

```
1 #include <stdio.h>
2 #include <unistd.h>
3 #include <ctype.h>
4 int main()
5 {
6     char string[100];
7     printf("Give string:");
8     scanf("%s", string);
9     int pld = fork();
10
11     if(pld == 0)
12     {
13         int vowel_count = 0;
14         for (int i = 0; string[i] != '\0'; i++) {
15             if (isalpha(string[i]) && (string[i] == 'a' || string[i] == 'e' || string[i] == 'i' || string[i] == 'o' || string[i] == 'u' || string[i] == 'A' || string[i] == 'E' || string[i] == 'I' || string[i] == 'O' || string[i] == 'U')) {
16                 vowel_count++;
17             }
18         }
19         printf("No of vowels: %d\n", vowel_count);
20     }
21     else
22     {
23         int consonant_count = 0;
24         for (int i = 0; string[i] != '\0'; i++) {
25             if (isalpha(string[i]) && !(string[i] == 'a' || string[i] == 'e' || string[i] == 'i' || string[i] == 'o' || string[i] == 'u' || string[i] == 'A' || string[i] == 'E' || string[i] == 'I' || string[i] == 'O' || string[i] == 'U')) {
26                 consonant_count++;
27             }
28         }
29         printf("No of consonants: %d\n", consonant_count);
30     }
31     return 0;
32 }
```

A terminal window titled 'spandan@spandan-VirtualBox' shows the execution of the program:

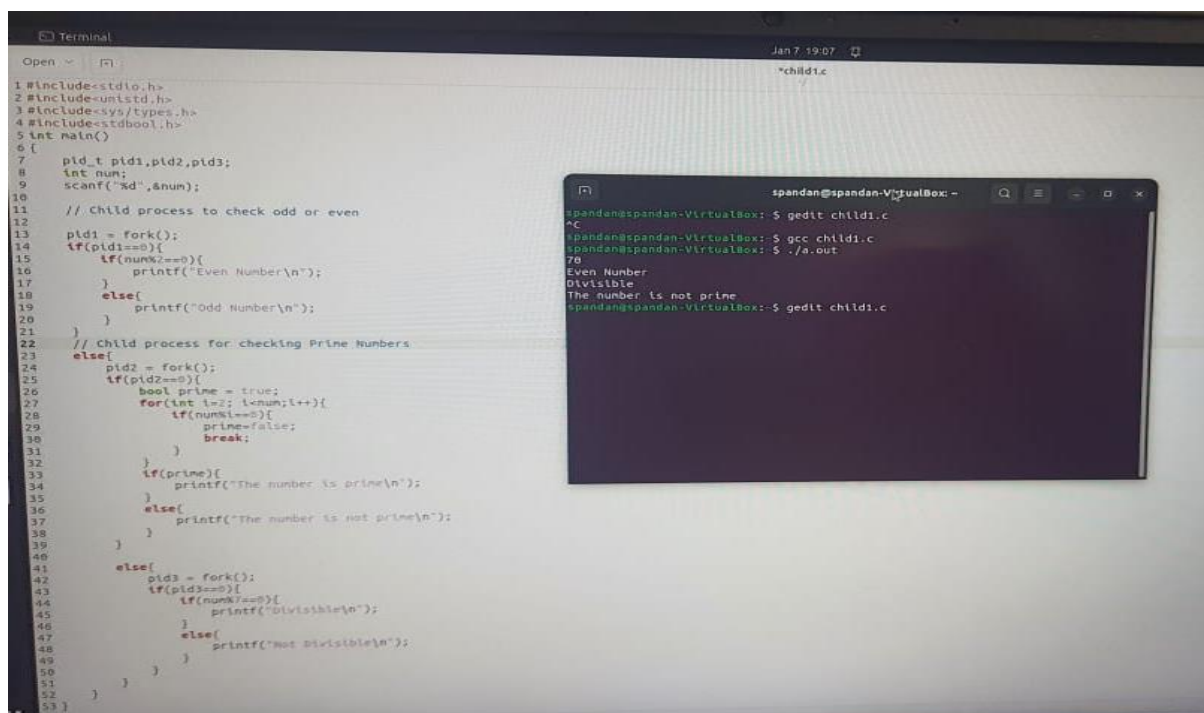
```
spandan@spandan-VirtualBox:~$ gcc vow.c
spandan@spandan-VirtualBox:~$ ./a.out
Give string:Spandan
No of vowels: 5
No of consonants: 2
```


- 5) Develop a child process and parent process to perform the matrix addition (by parent process) and matrix subtraction (by child process)



```
1 #include<stdio.h>
2 #include<sys/types.h>
3 #include<unistd.h>
4
5 int main()
6 {
7     int a[2][2] = {1,2,4,5};
8     int b[2][2] = {3,6,7,8};
9
10    int c[2][2], d[2][2];
11
12    int k = fork();
13
14    if(k==0)
15    {
16        for(int i=0;i<2;i++){
17            for(int j=0;j<2;j++){
18                c[i][j] = a[i][j]+b[i][j];
19                printf("%d\n",c[i][j]);
20            }
21        }
22    }
23
24    if(k>0)
25    {
26        for(int m=0;m<2;m++){
27            for(int n=0;n<2;n++){
28                d[m][n] = a[m][n]-b[m][n];
29                printf("%d\n",d[m][n]);
30            }
31        }
32    }
33
34    return 0;
35 }
36
37
38 }
```

- 6) Develop 3 child processes for doing the below tasks
Child process 1- check the given number is even or odd
Child process 2 - Check whether the given number is prime or not
child process 3 - Check whether the given number is divisible by 7 or not.



```
1 #include<stdio.h>
2 #include<unistd.h>
3 #include<sys/types.h>
4 #include<stdbool.h>
5 int main()
6 {
7     pid_t p1,p2,p3;
8     int num;
9     scanf("%d",&num);
10
11    // Child process to check odd or even
12
13    p1 = fork();
14    if(p1==0){
15        if(num%2==0){
16            printf("Even Number\n");
17        }
18        else{
19            printf("Odd Number\n");
20        }
21    }
22
23    // Child process for checking Prime Numbers
24    p2 = fork();
25    if(p2==0){
26        bool prime = true;
27        for(int i=2; i<num;i++){
28            if(num%i==0){
29                prime=false;
30                break;
31            }
32        }
33        if(prime){
34            printf("The number is prime\n");
35        }
36        else{
37            printf("The number is not prime\n");
38        }
39    }
40
41    p3 = fork();
42    if(p3==0){
43        if(num%7==0){
44            printf("Divisible\n");
45        }
46        else{
47            printf("Not Divisible\n");
48        }
49    }
50
51    return 0;
52 }
53 }
```