

```

1  /*
2      -----
3      SNAKE XENZIA
4      -----
5
6      ~~~~~
7      This is a basic snake game which is commonly
8      found in old nokia phones. This game has a
9      snake the player has to move with the arrow
10     keys. There will be food particles generated
11     on the screen from time to time. When the
12     snake eats them, its length increases. The
13     game ends when the snake bits its own body.
14     The player's objective is to get the highest
15     possible score by making the snake as long as
16     possible.
17
18     Features of this game:
19     1. 3 difficulty levels: Easy, Medium, Hard
20         Harder the difficulty level, faster will
21         be the movement of the snake.
22     2. The player can choose 3 different screen
23         sizes: Small, Medium, Large
24
25     ~~~~~
26     Created by: Anirudh Panigrahi
27                XI-E
28                Remal Public School, Sector-3
29                Rohini, New Delhi
30
31                Roll no.-11534
32
33     ~~~~~
34 */
35 #include <iostream.h>
36 #include <fstream.h>
37 #include <string.h>
38 #include <conio.h>
39 #include <dos.h>
40 #include <stdlib.h>
41
42 #define ARR_X 200
43 #define ARR_Y 200 //dimensions of pos_key matrix
44
45 char pos_key[ARR_X][ARR_Y]; //matrix corresponding to every position on the screen
46
47 void keystroke(); //function which receives and maps keystrokes and decides the corresponding
48 movement
49 void adjustxy(int&, int&); //helps in swapping screen sides when snake reaches one edge of screen
50 void movehead(int, int); //prints the head of the snake
51 void movetail(char [ARR_X][ARR_Y]); //replaces the last element of the snake with " "
52 void addfood(); //adds food particles in the game
53 int checkfood(); //checks if snake has eaten food or not
54 int checkdie(); //checks if snake has bitten itself or not
55 void printgameover(); //terminates game if checkdie() is true
56 char getdir(); //converts arrow key input to alphabets
57 void screen(); //sets all screen parameters at the beginning of the program
58 void frame(); //prints the borders of the screen
59 int pause(); //manages pause option
60 int select(int , int , int [], int); //helps in option selection in the game menus with arrow keys
61 void highscore(int); //to print and modify highscores
62
63 int MAX_X; //maximum horizontal coordinate of screen
64 int MIN_X; //minimum horizontal coordinate of screen
65 int MAX_Y; //maximum vertical coordinate of screen
66 int MIN_Y; //minimum vertical coordinate of screen
67 int max_xs; //MAX_X for small screen
68 int max_xm; //MAX_X for medium screen
69 int max_xl; //MAX_X for large screen
70 int max_ys; //MAX_X for small screen
71 int max_ym; //MAX_X for medium screen
72 int max_yl; //MAX_X for large screen
73 int x, y; //head coordinates
74 int x1, y1; //tail coordinates
75 int x2, y2; //food coordinates
76 int snaklen; //records length of the snake(to show score at end)
77 int frame_width = 150; //sets default game speed
78 int counter = 0; //records the no. of times the game is played
79
80 void main()
81 {
82     if(counter==0)
83         //FUNCTIONS TO RUN ONLY AT THE START OF THE PROGRAM//

```

```

81 {
82     randomize();    //to seed the random() function in addfood()
83     screen();
84 }
85 //WELCOME SCREEN//////////
86 int choice;
87 char choice2;
88 flag3:
89     snaklen = 1;
90 for(int l = 0; l<ARR_X; ++l)
91 {
92     for(int m = 0; m<ARR_Y; ++m)
93     {
94         pos_key[l][m] = 'k';    //to reset pos_key at start of every new game
95     }
96 }
97 clrscr();    frame();
98 _setcursortype(_NOCURS);
99 //MAIN MENU//////////
100 gotoxy((MAX_X - MIN_X + 1)/2 - 10, (MAX_Y - MIN_Y + 1)/2 - 6);
101 textcolor(YELLOW);
102 cprintf("*****");
103 gotoxy((MAX_X - MIN_X + 1)/2 - 10, (MAX_Y - MIN_Y + 1)/2 - 4);
104 cprintf("*****");
105 textcolor(MAGENTA);
106 cprintf("SNAKE XENZIA");
107 textcolor(YELLOW);
108 cprintf("*****");
109 gotoxy((MAX_X - MIN_X + 1)/2 - 10, (MAX_Y - MIN_Y + 1)/2 - 2);
110 cprintf("*****");
111 textcolor(WHITE);
112 gotoxy((MAX_X - MIN_X + 1)/2 - 10, (MAX_Y - MIN_Y + 1)/2);
113 cout<<"1. Play";
114 gotoxy((MAX_X - MIN_X + 1)/2 - 10, (MAX_Y - MIN_Y + 1)/2 + 1);
115 cout<<"2. Controls";
116 gotoxy((MAX_X - MIN_X + 1)/2 - 10, (MAX_Y - MIN_Y + 1)/2 + 2);
117 cout<<"3. Options";
118 gotoxy((MAX_X - MIN_X + 1)/2 - 10, (MAX_Y - MIN_Y + 1)/2 + 3);
119 cout<<"4. Highscores";
120 gotoxy((MAX_X - MIN_X + 1)/2 - 10, (MAX_Y - MIN_Y + 1)/2 + 4);
121 cout<<"5. Exit";
122 //END OF MAIN MENU//////////
123 int pos_gap1[12] = {1, 1, 1, 1, 1, -1};
124 choice = select((MAX_X - MIN_X + 1)/2 - 11, (MAX_Y - MIN_Y + 1)/2, pos_gap1, 5);
125 switch (choice)
126 {
127     case 1 :
128         clrscr();    frame();
129         x = (MAX_X - MIN_X + 1)/2, y = (MAX_Y - MIN_Y + 1)/2;
130         addfood();
131         keystroke();
132         break;
133     case 2 :
134         clrscr();    frame();
135         gotoxy((MAX_X - MIN_X + 1)/2 - 3, (MAX_Y - MIN_Y + 1)/2 - 3);
136         textcolor(YELLOW);
137         cprintf("CONTROLS");
138         textcolor(WHITE);
139         gotoxy((MAX_X - MIN_X + 1)/2 - 15, (MAX_Y - MIN_Y + 1)/2 - 1);
140         cout<<"1. Esc - Pauses the game";
141         gotoxy((MAX_X - MIN_X + 1)/2 - 15, (MAX_Y - MIN_Y + 1)/2 + 1);
142         cout<<"2. Arrow keys to move the snake";
143         gotoxy((MAX_X - MIN_X + 1)/2 - 18, (MAX_Y - MIN_Y + 1)/2 + 3);
144         cout<<"Press any key to return to main menu...";
145         getch();
146         goto flag3;
147     case 3 :
148         flag4:
149         clrscr();    frame();
150         //OPTIONS MENU//////////
151         gotoxy((MAX_X - MIN_X + 1)/2 - 3, (MAX_Y - MIN_Y + 1)/2 - 6);
152         textcolor(YELLOW);
153         cprintf("OPTIONS");
154         textcolor(WHITE);
155         gotoxy((MAX_X - MIN_X + 1)/2 - 8, (MAX_Y - MIN_Y + 1)/2 - 4);
156         cout<<"1. Set difficulty level: ";
157         gotoxy((MAX_X - MIN_X + 1)/2 - 7, (MAX_Y - MIN_Y + 1)/2 - 3);
158         cout<<"1.1 Easy";
159         gotoxy((MAX_X - MIN_X + 1)/2 - 7, (MAX_Y - MIN_Y + 1)/2 - 2);
160         cout<<"1.2 Medium";
161         gotoxy((MAX_X - MIN_X + 1)/2 - 7, (MAX_Y - MIN_Y + 1)/2 - 1);

```

```

162 cout<<"1.3 Hard";
163 gotoxy((MAX_X - MIN_X + 1)/2 - 8, (MAX_Y - MIN_Y + 1)/2 + 1);
164 cout<<"2. Set screen size";
165 gotoxy((MAX_X - MIN_X + 1)/2 - 7, (MAX_Y - MIN_Y + 1)/2 + 2);
166 cout<<"2.1 Small";
167 gotoxy((MAX_X - MIN_X + 1)/2 - 7, (MAX_Y - MIN_Y + 1)/2 + 3);
168 cout<<"2.2 Medium";
169 gotoxy((MAX_X - MIN_X + 1)/2 - 7, (MAX_Y - MIN_Y + 1)/2 + 4);
170 cout<<"2.3 Large";
171 gotoxy((MAX_X - MIN_X + 1)/2 - 8, (MAX_Y - MIN_Y + 1)/2 + 6);
172 cout<<"3. Back to main menu";
173 int pos_gap2[12] = {1, 1, 3, 3, 3, 1, 1, 2, 2, 1, -1};
174 choice = select((MAX_X - MIN_X + 1)/2 - 9, (MAX_Y - MIN_Y + 1)/2 - 3, pos_gap2, 10);
175 switch (choice)
176 {
177     case 1:
178         frame_width = 150;
179         gotoxy((MAX_X - MIN_X + 1)/2 - 16, (MAX_Y - MIN_Y + 1)/2 + 9);
180         textcolor(GREEN);
181         cprintf("The difficulty level is now set to 'Easy'");
182         textcolor(WHITE);
183         getch();
184         goto flag4;
185     case 2:
186         frame_width = 100;
187         gotoxy((MAX_X - MIN_X + 1)/2 - 16, (MAX_Y - MIN_Y + 1)/2 + 9);
188         textcolor(BLUE);
189         cprintf("The difficulty level is now set to 'Medium'");
190         textcolor(WHITE);
191         getch();
192         goto flag4;
193     case 3:
194         frame_width = 50;
195         gotoxy((MAX_X - MIN_X + 1)/2 - 16, (MAX_Y - MIN_Y + 1)/2 + 9);
196         textcolor(RED);
197         cprintf("The difficulty level is now set to 'Hard'");
198         textcolor(WHITE);
199         getch();
200         goto flag4;
201
202     case 6:
203         MAX_X = max_xs;
204         MAX_Y = max_ys;
205         clrscr(); frame();
206         gotoxy((MAX_X - MIN_X + 1)/2 - 16, (MAX_Y - MIN_Y + 1)/2 + 9);
207         textcolor(CYAN);
208         cprintf("The screen size is now set to 'Small'");
209         textcolor(WHITE);
210         getch();
211         goto flag4;
212     case 7:
213         MAX_X = max_xm;
214         MAX_Y = max_ym;
215         clrscr(); frame();
216         gotoxy((MAX_X - MIN_X + 1)/2 - 16, (MAX_Y - MIN_Y + 1)/2 + 9);
217         textcolor(CYAN);
218         cprintf("The screen size is now set to 'Medium'");
219         textcolor(WHITE);
220         getch();
221         goto flag4;
222     case 8:
223         MAX_X = max_xl;
224         MAX_Y = max_yl;
225         clrscr(); frame();
226         gotoxy((MAX_X - MIN_X + 1)/2 - 16, (MAX_Y - MIN_Y + 1)/2 + 9);
227         textcolor(CYAN);
228         cprintf("The screen size is now set to 'Large'");
229         textcolor(WHITE);
230         getch();
231         goto flag4;
232     case 10:
233         goto flag3;
234 }
235 break;
236 ///////////////////////////////////////////////////
237 case 4 :
238     highscore(0);
239     break;
240 case 5 :
241     clrscr(); frame();
242     gotoxy((MAX_X - MIN_X + 1)/2 - 15, (MAX_Y - MIN_Y + 1)/2 - 1);

```

```

243     textcolor(LIGHTRED);
244     cprintf("Are you sure you want to exit?");
245     gotoxy((MAX_X - MIN_X + 1)/2 - 15, (MAX_Y - MIN_Y + 1)/2 + 1);
246     cprintf("(hit key y or n...)");
247     textcolor(WHITE);
248     choice2 = getch();
249     if(choice2=='y' || choice2=='Y')
250     {
251         exit(0);
252     }
253     else
254     {
255         goto flag3;
256     }
257     break;
258 }
259 goto flag3;
260 //END OF WELCOME SCREEN////
261 //END OF MAIN////
262 }
263
264 void keystroke()
265 {
266     flag:
267     movehead(x, y);           //to show initial direction of the snake
268     char c0, c = getdir();
269     //TO SET INITIAL TAIL COORDINATES//
270     switch(c)
271     {
272         case 'w':
273             x1 = x;
274             y1 = y+1;
275             break;
276         case 's':
277             x1 = x;
278             y1 = y-1;
279             break;
280         case 'a':
281             x1 = x+1;
282             y1 = y;
283             break;
284         case 'd':
285             x1 = x-1;
286             y1 = y;
287             break;
288         case 27 :
289             if(pause())
290             {
291                 return; //gives the option of exiting the game before starting to play
292             }
293             else
294             {
295                 goto flag;
296             }
297         default :
298             goto flag;
299     }
300     pos_key[x1][y1] = c;
301     //LOOPS TO SET HEAD COORDINATES, MAP THE DIRECTION VALUE OF THE HEAD//
302     //CORRESPONDING TO ITS POSITION AND CALL MOVEHEAD() AND MOVETAIL() //
303     flag2:
304     switch(c)
305     {
306         case 'w':
307             do
308             {
309                 y--;
310                 adjustxy(x, y);
311                 movehead(x, y);
312                 y==MAX_Y ? pos_key[x][MIN_Y] : pos_key[x][y+1] = 'w';
313                 movetail(pos_key);
314                 printgameover();
315                 delay(frame_width);
316             }while(!kbhit());
317             break;
318         case 's':
319             do
320             {
321                 y++;
322                 adjustxy(x, y);
323                 movehead(x, y);

```

```

324         y==MIN_Y ? pos_key[x][MAX_Y] : pos_key[x][y-1] = 's';
325         movetail(pos_key);
326         printgameover();
327         delay(frame_width);
328     }while(!kbhit());
329     break;
330 case 'a':
331     do
332     {
333         x--;
334         adjustxy(x, y);
335         movehead(x, y);
336         x==MAX_X ? pos_key[MIN_X][y] : pos_key[x+1][y] = 'a';
337         movetail(pos_key);
338         printgameover();
339         delay(frame_width);
340     }while(!kbhit());
341     break;
342 case 'd':
343     do
344     {
345         x++;
346         adjustxy(x, y);
347         movehead(x, y);
348         x==MIN_X ? pos_key[MAX_X][y] : pos_key[x-1][y] = 'd';
349         movetail(pos_key);
350         printgameover();
351         delay(frame_width);
352     }while(!kbhit());
353     break;
354 }
355 c0 = getdir();
356 //TO IGNORE OPPOSITE DIRECTION KEYSTROKE AND ANY OTHER KEYSTROKE//
357 if(c0==27)
358 {
359     if(pause())
360     {
361         return;
362     }
363     else
364     {
365         goto flag2;
366     }
367 }
368 else
369 {
370     switch (c)
371     {
372     case 'w':
373         if (c0=='a' || c0=='d')
374         {
375             c = c0;
376             goto flag2;
377         }
378         else
379         {
380             goto flag2;
381         }
382     case 's':
383         if (c0=='a' || c0=='d')
384         {
385             c = c0;
386             goto flag2;
387         }
388         else
389         {
390             goto flag2;
391         }
392     case 'a':
393         if (c0=='w' || c0=='s')
394         {
395             c = c0;
396             goto flag2;
397         }
398         else
399         {
400             goto flag2;
401         }
402     case 'd':
403         if (c0=='w' || c0=='s')
404         {

```

```

405         c = c0;
406         goto flag2;
407     }
408     else
409     {
410         goto flag2;
411     }
412 }
413 }
414
415 }
416 void adjustxy(int &x, int &y)
417 {
418     if (y == MAX_Y + 1)
419         y = MIN_Y;
420     else if (y == MIN_Y - 1)
421         y = MAX_Y;
422     if (x == MAX_X + 1)
423         x = MIN_X;
424     else if (x == MIN_X - 1)
425         x = MAX_X;
426     // gotoxy(x, y);
427     // cout<<"@"; //<<x<<"", "<<y;    //for testing purposes
428     return;
429 }
430 void movehead (int x, int y)
431 {
432     gotoxy(x, y);
433     textcolor(WHITE);
434     cprintf("@");
435     // delay(500);    //for testing purposes
436 }
437
438 void movetail (char pos_key[ARR_X][ARR_Y])
439 {
440     if(checkfood()!=0)
441     {
442         ++snaklen;
443         addfood();
444         return;
445     }
446     else if (checkfood()==0)
447     {
448         //TO SET THE NEXT TAIL COORDINATES ACCORDING TO THE DIRECTION //
449         //VALUE STORED IN POS_KEY AT THE EXISTING POSITION OF THE TAIL//
450         switch(pos_key[x1][y1])
451         {
452             case 'w':
453                 pos_key[x1][y1] = ' ';
454                 y1--;
455                 adjustxy(x1, y1);
456                 gotoxy(x1, y1);
457                 cout<<" ";
458                 break;
459             case 's':
460                 pos_key[x1][y1] = ' ';
461                 y1++;
462                 adjustxy(x1, y1);
463                 gotoxy(x1, y1);
464                 cout<<" ";
465                 break;
466             case 'a':
467                 pos_key[x1][y1] = ' ';
468                 x1--;
469                 adjustxy(x1, y1);
470                 gotoxy(x1, y1);
471                 cout<<" ";
472                 break;
473             case 'd':
474                 pos_key[x1][y1] = ' ';
475                 x1++;
476                 adjustxy(x1, y1);
477                 gotoxy(x1, y1);
478                 cout<<" ";
479                 break;
480         }
481     }
482 }
483 }
484 // delay(500);    //for testing purposes
485 return;

```

```

486
487
488 }
489 void addfood()
490 {
491     //LOOP WILL RUN UNTIL X2, Y2 ARE SET TO VALUES WHICH ARE NOT ON THE SNAKE//
492     do
493     {
494         x2 = random(MAX_X - MIN_X + 1) + MIN_X;
495         y2 = random(MAX_Y - MIN_Y + 1) + MIN_Y;
496     }while(x2==x && y2==y || (pos_key[x2][y2]=='w' || pos_key[x2][y2]=='s' ||
497         pos_key[x2][y2]=='a' || pos_key[x2][y2]=='d' ));
498     gotoxy(x2, y2);
499     textcolor(LIGHTMAGENTA);
500     cprintf("@");
501     textcolor(WHITE);
502     return;
503 }
504
505 int checkfood()
506 {
507     if(x==x2 && y==y2)
508     {
509         return 1;
510     }
511     else
512     {
513         return 0;
514     }
515 }
516
517 int checkdie()
518 {
519     //IF POS_KEY[X][Y] IS ALREADY MAPPED WITH A///
520     //DIRECTION VALUE WHEN THE HEAD REACHES (X, Y) COORDINATES///
521     if(pos_key[x][y]=='w' || pos_key[x][y]=='s' ||
522        pos_key[x][y]=='a' || pos_key[x][y]=='d' )
523     {
524         return 1;
525     }
526     else
527     {
528         return 0;
529     }
530 }
531
532 void printgameover()
533 {
534     if(checkdie()!=0)
535     {
536         int k = 1;
537         delay(frame_width);
538         //LOOP TO FLASH "GAME OVER" 4 TIMES//
539         do
540         {
541             textcolor(RED);
542             gotoxy((MAX_X - MIN_X + 1)/2 - 5 , (MAX_Y - MIN_Y + 1)/2);
543             cprintf("GAME OVER!!\a");
544             textcolor(WHITE);
545             delay(400);
546             clrscr(); frame();
547             delay(400);
548             ++k;
549         }while(k<=4);
550         while(kbhit()) //to ignore keystrokes pressed while displaying game over//
551         { //so that final score can be visible//
552             getch();
553         }
554         textcolor(RED);
555         gotoxy((MAX_X - MIN_X + 1)/2 - 5 , (MAX_Y - MIN_Y + 1)/2);
556         cprintf("GAME OVER!!");
557         textcolor(WHITE);
558         gotoxy((MAX_X - MIN_X + 1)/2 - 8 , (MAX_Y - MIN_Y + 1)/2 + 1);
559         cout<<"Final Score: "<<snaklen*10;
560         gotoxy((MAX_X - MIN_X + 1)/2 - 12 , (MAX_Y - MIN_Y + 1)/2 + 2);
561         cout<<"Press any key to exit...";
562         getch();
563         while(kbhit()) { getch(); }
564         highscore(snaklen*10);
565         ++counter;
566         main();
567     }
568 }

```

```

567 char getdir()
568 {
569     char ch = getch();
570     if(ch==0)
571     {
572         ch = getch();
573         switch(ch)
574         {
575             case 'H':
576                 return 'w';
577             case 'P':
578                 return 's';
579             case 'K':
580                 return 'a';
581             case 'M':
582                 return 'd';
583             default :
584                 return 'x';
585         }
586     }
587     else if(ch==27)
588     {
589         return ch;
590     }
591     else
592     {
593         return 'x';    //any random character, so that it can be ignored by keystroke()
594     }
595 }
596 void screen()
597 {
598     struct text_info info;
599     gettextinfo(&info);
600
601     MAX_X = (int) info.winright - 1;
602     MIN_X = (int) info.winleft + 1;
603     MAX_Y = (int) info.winbottom - 2;
604     MIN_Y = (int) info.wintop + 1;
605     max_xs = (int) MAX_X * 0.8;
606     max_xm = (int) MAX_X * 0.9;
607     max_xl = (int) MAX_X;
608     max_ys = (int) MAX_Y * 0.8;
609     max_ym = (int) MAX_Y * 0.9;
610     max_yl = (int) MAX_Y;
611     frame();
612     return;
613 }
614 void frame()
615 {
616
617     int width = MAX_X - MIN_X + 3;
618     int height = MAX_Y - MIN_Y + 4;
619     textcolor(YELLOW);
620     gotoxy(1,1);
621     for(int i = 0; i < width; i++)
622     {
623         cprintf("%c", ' ');
624     }
625
626     for(i = 2; i <= height - 2; i++)
627     {
628         gotoxy(1, i); cprintf ("%c", ' ');
629         gotoxy(width, i); cprintf ("%c", ' ');
630     }
631
632     gotoxy(1, height - 1);
633     for(i = 0; i < width; i++)
634     {
635         cprintf ("%c", ' ');
636     }
637     textcolor(WHITE);
638 }
639 int pause()
640 {
641     textcolor(GREEN);
642     gotoxy((MAX_X-MIN_X+1)/2 - 15, MIN_Y - 1);
643     cprintf("PAUSED*Press esc again to exit");
644     gotoxy((MAX_X-MIN_X+1)/2 - 15, MAX_Y + 1);
645     cprintf("Press any other key to resume");
646     textcolor(WHITE);
647     char ch = getch();

```



```

648     frame();
649     if(ch==27)
650     {
651         return 1;
652     }
653     else
654     {
655         return 0;
656     }
657 }
658 int select(int x_opt, int y_opt, int pos_gap[], int totalgap)
659 //X_OPT - X COORDINATE OF THE BULLET IN ALL POSITIONS//
660 //Y_OPT - Y COORDINATE OF BULLET FOR FIRST OPTION//
661 //POS_GAP[Y] - THE GAP B/W THE OPTION AT (Y+1) COORDINATE AND THE NEXT OPTION TO IT//
662 //TOTALGAP - THE TOTAL LINES OCCUPIED BY ALL OPTIONS//
663 {
664     char ch = 26;
665     int y_init = y_opt;
666     int x_init = x_opt;
667     gotoxy(x_init, y_opt);
668     cprintf("%c", ch);
669     do
670     {
671         //gotoxy(2, 2);cout<<x_init<<" "<<y_init<<" "
672         //cout<<x_opt<<" "<<y_opt<<" "<<pos_gap[y_opt-y_init]; //for testing purposes
673         char c = getch();
674         if(c==0)
675         {
676             c = getch();
677             gotoxy(x_init, y_opt);
678             cout<<" "; //to delete initial bullet
679             //TO SET Y COORDINATE OF NEW BULLET ACCORDING TO ARROW KEY PRESSED//
680             switch(c)
681             {
682                 case 'H' :
683                     if(y_opt==y_init)
684                     {
685                         y_opt = y_init + totalgap - 1;
686                     }
687                     else
688                     {
689                         y_opt = y_opt - pos_gap[y_opt-y_init - 1];
690                     }
691                     break;
692                 case 'P' :
693                     if(y_opt==(y_init + totalgap - 1))
694                     {
695                         y_opt = y_init;
696                     }
697                     else
698                     {
699                         y_opt = y_opt + pos_gap[y_opt-y_init];
700                     }
701                     break;
702             }
703             gotoxy(x_init, y_opt); //to print new bullet
704             cprintf("%c", ch);
705         }
706         else if(c==13)
707         {
708             return y_opt - y_init + 1; //to return option value according
709                                     //to y coordinate of bullet
710         }
711     }while(1);
712 }
713 void highscore(int s)
714 {
715     struct player_score
716     {
717         int difficulty;
718         int score;
719         char name[9];
720     }p_s[9];
721     ////INITIALIZING STRUCTURE TO AVOID GARBAGE VALUES////
722     for(int i = 0; i<9; ++i)
723     {
724         strcpy(p_s[i].name, " ");
725         p_s[i].score = 0;
726     }
727     for(i = 0; i<9; ++i)
728     {

```

```

729     if(i<3)
730         p_s[i].difficulty = 150;
731     else if(i>=3 && i<6)
732         p_s[i].difficulty = 100;
733     else if(i>=6 && i<9)
734         p_s[i].difficulty = 50;
735 }
736 ///HIGHSCORES WILL BE STORED IN BINARY FILE 'HIGHSCOR'///
737 ifstream finout;
738 finout.open("HIGHSCOR", ios::in | ios::nocreate | ios::binary);
739 ///TO CREATE 'HIGHSCOR' FILE IF IT IS NOT PRESENT///
740 if(finout == 0)
741 {
742     ofstream ftemp;
743     ftemp.open("HIGHSCOR", ios::out | ios::binary);
744     for(int i = 0; i < 9; ++i)
745     {
746         ftemp.write((char *) &p_s[i], sizeof(player_score));
747     }
748     ftemp.close();
749     ifstream finout;
750     finout.open("HIGHSCOR", ios::in | ios::nocreate | ios::binary);
751 }
752 for(i = 0; i < 9; ++i)
753 {
754     finout.read((char *) &p_s[i] , sizeof(player_score));
755 }
756 finout.close();
757 int line_no, j = 2;
758 if(s != 0)
759 {
760     int j = 2;
761     while(p_s[j].difficulty != frame_width)
762     {
763         j = j + 3;
764     }
765     for(int i = j - 2; s < p_s[i].score; ++i){}
766     if(i>j){ return; }
767     line_no = i;
768     while(j > line_no)
769     {
770         p_s[j] = p_s[j-1];
771         --j;
772     }
773     p_s[line_no].score = s;
774 }
775 //////////////////////////////////HIGHSCORES PAGE////////////////////////////////
776 clrscr(); frame();
777 gotoxy((MAX_X - MIN_X + 1)/2 - 5, (MAX_Y - MIN_Y + 1)/2 - 7);
778 textcolor(YELLOW);
779 cprintf("HIGHSCORES");
780 textcolor(WHITE);
781 gotoxy((MAX_X - MIN_X + 1)/2 - 17, (MAX_Y - MIN_Y + 1)/2 - 5);
782 cout<<"EASY: ";
783 for(i = 0; i < 3; ++i)
784 {
785     gotoxy((MAX_X - MIN_X + 1)/2 - 10, (MAX_Y - MIN_Y + 1)/2 - 5 + i);
786     cout<<i+1<<" ";
787     for(int j = 0; j<8; ++j)
788         cout.put(p_s[i].name[j]);
789     cout<<" "<<p_s[i].score;
790 }
791 gotoxy((MAX_X - MIN_X + 1)/2 - 19, (MAX_Y - MIN_Y + 1)/2 - 1);
792 cout<<"MEDIUM: ";
793 for(i = 3; i < 6; ++i)
794 {
795     gotoxy((MAX_X - MIN_X + 1)/2 - 10, (MAX_Y - MIN_Y + 1)/2 - 4 + i);
796     cout<<i+1<<" ";
797     for(int j = 0; j<8; ++j)
798         cout.put(p_s[i].name[j]);
799     cout<<" "<<p_s[i].score;
800 }
801 gotoxy((MAX_X - MIN_X + 1)/2 - 17, (MAX_Y - MIN_Y + 1)/2 + 3);
802 cout<<"HARD: ";
803 for(i = 6; i < 9; ++i)
804 {
805     gotoxy((MAX_X - MIN_X + 1)/2 - 10, (MAX_Y - MIN_Y + 1)/2 - 3 + i);
806     cout<<i+1<<" ";
807     for(int j = 0; j<8; ++j)
808         cout.put(p_s[i].name[j]);
809     cout<<" "<<p_s[i].score;

```

```

810 }
811 if(s == 0)
812 {
813     finout.close();
814     gotoxy((MAX_X - MIN_X + 1)/2 - 10, (MAX_Y - MIN_Y + 1)/2 + 7);
815     cout<<"Press any key to go back to main menu...";
816     getch();
817 }
818 else
819 {
820     gotoxy((MAX_X - MIN_X + 1)/2 - 10, (MAX_Y - MIN_Y + 1)/2 + 7);
821     cout<<"Enter your name...";
822     _setcursortype(_SOLIDCURSOR);
823     char name[8];
824     for(int i = 0; p_s[i].difficulty != frame_width; i = i + 3){}
825     gotoxy((MAX_X - MIN_X + 1)/2 - 7, line_no + (i / 3) + (MAX_Y - MIN_Y + 1)/2 - 5);
826     cout<<"          ";
827     gotoxy((MAX_X - MIN_X + 1)/2 - 7, line_no + (i / 3) + (MAX_Y - MIN_Y + 1)/2 - 5);
828     cin.get(name, 9);
829     cin.ignore(1000, '\n');
830     strcpy(p_s[line_no].name, "          ");
831     strcpy(p_s[line_no].name, name);
832     _setcursortype(_NOCURSOR);
833     ofstream finout;
834     finout.open("HIGHSCOR", ios::out | ios::binary);
835     for(i = 0; i < 9; ++i)
836     {
837         finout.write((char *) &p_s[i] , sizeof(player_score));
838     }
839     finout.close();
840 }
841 }
842 }

```