```
/*
            ------------
            SNAKE XENZIA
            ------------
    ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    This is a basic snake game which is commonly
    found in old nokia phones. This game has a
    snake the player has to move with the arrow
    keys. There will be food particles generated
    on the screen from time to time. When the
    snake eats them, its length increases. The
    game ends when the snake bits its own body.
    The player's objective is to get the highest
    possible score by making the snake as long as
    possible.

    Features of this game:
    1. 3 difficulty levels: Easy, Medium, Hard
       Harder the difficulty level, faster will
       be the movement of the snake.
    2. The player can choose 3 different screen
       sizes: Small, Medium, Large

    ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    Created by: Anirudh Panigrahi
            XI-E
            Remal Public School, Sector-3
            Rohini, New Delhi

            Roll no.-11534
    ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
*/
#include <iostream.h>
#include <conio.h>
#include <dos.h>
#include <stdlib.h>

#define ARR_X 200
#define ARR_Y 200   //dimensions of pos_key matrix

char pos_key[ARR_X][ARR_Y]; //matrix corresponding to every position on the screen

void keystroke();           //function which receives and maps keystrokes and decides the corresponding movement
void adjustxy(int&, int&);  //helps in swapping screen sides when snake reaches one edge of screen
void movehead(int, int);    //prints the head of the snake
void movetail(char [ARR_X][ARR_Y]); //replaces the last element of the snake with " "
void addfood();             //adds food particles in the game
int checkfood();            //checks if snake has eaten food or not
int checkdie();             //checks if snake has bitten itself or not
void printgameover();       //terminates game if checkdie() is true
char getdir();              //converts arrow key input to alphabets
void screen();              //sets all screen parameters at the beginning of the program
void frame();               //prints the borders of the screen
int pause();                //manages pause option
int select(int , int , int [], int);    //helps in option selection in the game menus with arrow keys

int MAX_X;          //maximum horizontal coordinate of screen
int MIN_X;          //minimum horizontal coordinate of screen
int MAX_Y;          //maximum vertical coordinate of screen
int MIN_Y;          //minimum vertical coordinate of screen
int max_xs;         //MAX_X for small screen
int max_xm;         //MAX_X for medium screen
int max_xl;         //MAX_X for large screen
int max_ys;         //MAX_X for small screen
int max_ym;         //MAX_X for medium screen
int max_yl;         //MAX_X for large screen
int x, y;               //head coordinates
int x1, y1;             //tail coordinates
int x2, y2;             //food coordinates
int snaklen;        //records length of the snake(to show score at end)
int frame_width = 150;  //sets default game speed
int counter = 0;        //records the no. of times the game is played

void main()
{
    if(counter==0)
    //FUNCTIONS TO RUN ONLY AT THE START OF THE PROGRAM//
    {
        randomize();    //to seed the random() function in addfood()
        screen();
```

```
    }
///////WELCOME SCREEN/////////
    int choice;
    char choice2;
    flag3:
    snaklen = 1;
    for(int l = 0; l<ARR_X; ++l)
    {
        for(int m = 0; m<ARR_Y; ++m)
        {
            pos_key[l][m] = 'k';        //to reset pos_key at start of every new game
        }
    }
    clrscr();   frame();
    _setcursortype(_NOCURSOR);
    ////////MAIN MENU/////////
    gotoxy((MAX_X - MIN_X + 1)/2 - 10, (MAX_Y - MIN_Y + 1)/2 - 6);
    textcolor(YELLOW);
    cprintf("*******************");
    gotoxy((MAX_X - MIN_X + 1)/2 - 10, (MAX_Y - MIN_Y + 1)/2 - 4);
    cprintf("****");
    textcolor(MAGENTA);
    cprintf("SNAKE XENZIA");
    textcolor(YELLOW);
    cprintf("****");
    gotoxy((MAX_X - MIN_X + 1)/2 - 10, (MAX_Y - MIN_Y + 1)/2 - 2);
    cprintf("*******************");
    textcolor(WHITE);
    gotoxy((MAX_X - MIN_X + 1)/2 - 10, (MAX_Y - MIN_Y + 1)/2);
    cout<<"1. Play";
    gotoxy((MAX_X - MIN_X + 1)/2 - 10, (MAX_Y - MIN_Y + 1)/2 + 1);
    cout<<"2. Controls";
    gotoxy((MAX_X - MIN_X + 1)/2 - 10, (MAX_Y - MIN_Y + 1)/2 + 2);
    cout<<"3. Options";
    gotoxy((MAX_X - MIN_X + 1)/2 - 10, (MAX_Y - MIN_Y + 1)/2 + 3);
    cout<<"4. Exit";
    /////////END OF MAIN MENU/////////
    int pos_gap1[12] = {1, 1, 1, 1, -1};
    choice = select((MAX_X - MIN_X + 1)/2 - 11, (MAX_Y - MIN_Y + 1)/2, pos_gap1, 4);
    switch (choice)
    {
        case 1 :
            clrscr();   frame();
            x = (MAX_X - MIN_X + 1)/2, y = (MAX_Y - MIN_Y + 1)/2;
            addfood();
            keystroke();
            break;
        case 2 :
            clrscr();   frame();
            gotoxy((MAX_X - MIN_X + 1)/2 - 3, (MAX_Y - MIN_Y + 1)/2 - 3);
            textcolor(YELLOW);
            cprintf("CONTROLS");
            textcolor(WHITE);
            gotoxy((MAX_X - MIN_X + 1)/2 - 15, (MAX_Y - MIN_Y + 1)/2 - 1);
            cout<<"1. Esc - Pauses the game";
            gotoxy((MAX_X - MIN_X + 1)/2 - 15, (MAX_Y - MIN_Y + 1)/2 + 1);
            cout<<"2. Arrow keys to move the snake";
            gotoxy((MAX_X - MIN_X + 1)/2 - 18, (MAX_Y - MIN_Y + 1)/2 + 3);
            cout<<"Press any key to return to main menu...";
            getch();
            goto flag3;
        case 3 :
            flag4:
            clrscr();   frame();
            ////////OPTIONS MENU////////
            gotoxy((MAX_X - MIN_X + 1)/2 - 3, (MAX_Y - MIN_Y + 1)/2 - 6);
            textcolor(YELLOW);
            cprintf("OPTIONS");
            textcolor(WHITE);
            gotoxy((MAX_X - MIN_X + 1)/2 - 8, (MAX_Y - MIN_Y + 1)/2 - 4);
            cout<<"1. Set difficulty level: ";
            gotoxy((MAX_X - MIN_X + 1)/2 - 7, (MAX_Y - MIN_Y + 1)/2 - 3);
            cout<<"1.1 Easy";
            gotoxy((MAX_X - MIN_X + 1)/2 - 7, (MAX_Y - MIN_Y + 1)/2 - 2);
            cout<<"1.2 Medium";
            gotoxy((MAX_X - MIN_X + 1)/2 - 7, (MAX_Y - MIN_Y + 1)/2 - 1);
            cout<<"1.3 Hard";
            gotoxy((MAX_X - MIN_X + 1)/2 - 8, (MAX_Y - MIN_Y + 1)/2 + 1);
            cout<<"2. Set screen size";
            gotoxy((MAX_X - MIN_X + 1)/2 - 7, (MAX_Y - MIN_Y + 1)/2 + 2);
            cout<<"2.1 Small";
            gotoxy((MAX_X - MIN_X + 1)/2 - 7, (MAX_Y - MIN_Y + 1)/2 + 3);
            cout<<"2.2 Medium";
            gotoxy((MAX_X - MIN_X + 1)/2 - 7, (MAX_Y - MIN_Y + 1)/2 + 4);
            cout<<"2.3 Large";
            gotoxy((MAX_X - MIN_X + 1)/2 - 8, (MAX_Y - MIN_Y + 1)/2 + 6);
            cout<<"3. Back to main menu";
            int pos_gap2[12] = {1, 1, 3, 3, 3, 1, 1, 2, 2, 1, -1};
            choice = select((MAX_X - MIN_X + 1)/2 - 9, (MAX_Y - MIN_Y + 1)/2 - 3, pos_gap2, 10);
            switch (choice)
            {
                case 1:
                    frame_width = 150;
                    gotoxy((MAX_X - MIN_X + 1)/2 - 16, (MAX_Y - MIN_Y + 1)/2 + 9);
                    textcolor(GREEN);
                    cprintf("The difficulty level is now set to 'Easy'");
                    textcolor(WHITE);
                    getch();
                    goto flag4;
                case 2:
                    frame_width = 100;
                    gotoxy((MAX_X - MIN_X + 1)/2 - 16, (MAX_Y - MIN_Y + 1)/2 + 9);
                    textcolor(BLUE);
                    cprintf("The difficulty level is now set to 'Medium'");
                    textcolor(WHITE);
                    getch();
                    goto flag4;
                case 3:
                    frame_width = 50;
                    gotoxy((MAX_X - MIN_X + 1)/2 - 16, (MAX_Y - MIN_Y + 1)/2 + 9);
                    textcolor(RED);
                    cprintf("The difficulty level is now set to 'Hard'");
                    textcolor(WHITE);
                    getch();
                    goto flag4;

                case 6:
                    MAX_X = max_xs;
                    MAX_Y = max_ys;
                    clrscr();   frame();
                    gotoxy((MAX_X - MIN_X + 1)/2 - 16, (MAX_Y - MIN_Y + 1)/2 + 9);
                    textcolor(CYAN);
                    cprintf("The screen size is now set to 'Small'");
                    textcolor(WHITE);
                    getch();
                    goto flag4;
                case 7:
                    MAX_X = max_xm;
                    MAX_Y = max_ym;
                    clrscr();   frame();
                    gotoxy((MAX_X - MIN_X + 1)/2 - 16, (MAX_Y - MIN_Y + 1)/2 + 9);
                    textcolor(CYAN);
                    cprintf("The screen size is now set to 'Medium'");
                    textcolor(WHITE);
                    getch();
                    goto flag4;
                case 8:
                    MAX_X = max_xl;
                    MAX_Y = max_yl;
                    clrscr();   frame();
                    gotoxy((MAX_X - MIN_X + 1)/2 - 16, (MAX_Y - MIN_Y + 1)/2 + 9);
                    textcolor(CYAN);
                    cprintf("The screen size is now set to 'Large'");
                    textcolor(WHITE);
                    getch();
                    goto flag4;
                case 10:
                    goto flag3;
            }
            break;
            ////////END OF OPTIONS MENU/////////
        case 4 :
            clrscr();   frame();
            gotoxy((MAX_X - MIN_X + 1)/2 - 15, (MAX_Y - MIN_Y + 1)/2 - 1);
            textcolor(LIGHTRED);
            cprintf("Are you sure you want to exit?");
            gotoxy((MAX_X - MIN_X + 1)/2 - 15, (MAX_Y - MIN_Y + 1)/2 + 1);
            cprintf("(hit key y or n...)");
            textcolor(WHITE);
            choice2 = getch();
            if(choice2=='y' || choice2=='Y')
            {
```

```c
                exit(0);
            }
            else
            {
                goto flag3;
            }
            break;
    }
    goto flag3;
  /////END OF WELCOME SCREEN/////
  /////END OF MAIN/////
}

void keystroke()
{
    flag:
    movehead(x, y);         //to show initial direction of the snake
    char c0, c = getdir();
    //TO SET INITIAL TAIL COORDINATES//
    switch(c)
    {
        case 'w':
            x1 = x;
            y1 = y+1;
            break;
        case 's':
            x1 = x;
            y1 = y-1;
            break;
        case 'a':
            x1 = x+1;
            y1 = y;
            break;
        case 'd':
            x1 = x-1;
            y1 = y;
            break;
        case 27 :
            if(pause())
            {
                return; //gives the option of exiting the game before starting to play
            }
            else
            {
                goto flag;
            }
        default :
            goto flag;
    }
    pos_key[x1][y1] = c;
    //LOOPS TO SET HEAD COORDINATES, MAP THE DIRECTION VALUE OF THE HEAD//
    //CORRESPONDING TO ITS POSITION AND CALL MOVEHEAD() AND MOVETAIL()  //
    flag2:
        switch(c)
        {
            case 'w':
                do
                {
                    y--;
                    adjustxy(x, y);
                    movehead(x, y);
                    y==MAX_Y ? pos_key[x][MIN_Y] : pos_key[x][y+1] = 'w';
                    movetail(pos_key);
                    printgameover();
                    delay(frame_width);
                }while(!kbhit());
                break;
            case 's':
                do
                {
                    y++;
                    adjustxy(x, y);
                    movehead(x, y);
                    y==MIN_Y ? pos_key[x][MAX_Y] : pos_key[x][y-1] = 's';
                    movetail(pos_key);
                    printgameover();
                    delay(frame_width);
                }while(!kbhit());
                break;
            case 'a':
                do
```

```
                    {
                        x--;
                        adjustxy(x, y);
                        movehead(x, y);
                        x==MAX_X ? pos_key[MIN_X][y] : pos_key[x+1][y] = 'a';
                        movetail(pos_key);
                        printgameover();
                        delay(frame_width);
                    }while(!kbhit());
                    break;
                case 'd':
                    do
                    {
                        x++;
                        adjustxy(x, y);
                        movehead(x, y);
                        x==MIN_X ? pos_key[MAX_X][y] : pos_key[x-1][y] = 'd';
                        movetail(pos_key);
                        printgameover();
                        delay(frame_width);
                    }while(!kbhit());
                    break;
            }
            c0 = getdir();
            //TO IGNORE OPPOSITE DIRECTION KEYSTROKE AND ANY OTHER KEYSTROKE//
            if(c0==27)
            {
                if(pause())
                {
                    return;
                }
                else
                {
                    goto flag2;
                }
            }
            else
            {
                switch (c)
                {
                    case 'w':
                        if (c0=='a' || c0=='d')
                        {
                            c = c0;
                            goto flag2;
                        }
                        else
                        {
                            goto flag2;
                        }
                    case 's':
                        if (c0=='a' || c0=='d')
                        {
                            c = c0;
                            goto flag2;
                        }
                        else
                        {
                            goto flag2;
                        }
                    case 'a':
                        if (c0=='w' || c0=='s')
                        {
                            c = c0;
                            goto flag2;
                        }
                        else
                        {
                            goto flag2;
                        }
                    case 'd':
                        if (c0=='w' || c0=='s')
                        {
                            c = c0;
                            goto flag2;
                        }
                        else
                        {
                            goto flag2;
                        }
                }
```

```
        }

}
void adjustxy(int &x, int &y)
{
    if (y == MAX_Y + 1)
        y = MIN_Y;
    else if (y == MIN_Y - 1)
        y = MAX_Y;
    if (x == MAX_X + 1)
        x = MIN_X;
    else if (x == MIN_X - 1)
        x = MAX_X;
//  gotoxy(x, y);
//  cout<<"@";//<<x<<", "<<y;   //for testing purposes
    return;
}
void movehead (int x, int y)
{
    gotoxy(x, y);
    textcolor(WHITE);
    cprintf("@");
//  delay(500);                //for testing purposes
}

void movetail (char pos_key[ARR_X][ARR_Y])
{
    if(checkfood()!=0)
    {
        ++snaklen;
        addfood();
        return;
    }
    else if (checkfood()==0)
    {
    //TO SET THE NEXT TAIL COORDINATES ACCORDING TO THE DIRECTION //
    //VALUE STORED IN POS_KEY AT THE EXISTING POSITION OF THE TAIL//
        switch(pos_key[x1][y1])
        {
            case 'w':
                pos_key[x1][y1] = ' ';
                y1--;
                adjustxy(x1, y1);
                gotoxy(x1, y1);
                cout<<" ";
                break;
            case 's':
                pos_key[x1][y1] = ' ';
                y1++;
                adjustxy(x1, y1);
                gotoxy(x1, y1);
                cout<<" ";
                break;
            case 'a':
                pos_key[x1][y1] = ' ';
                x1--;
                adjustxy(x1, y1);
                gotoxy(x1, y1);
                cout<<" ";
                break;
            case 'd':
                pos_key[x1][y1] = ' ';
                x1++;
                adjustxy(x1, y1);
                gotoxy(x1, y1);
                cout<<" ";
                break;

        }

    }
//      delay(500);    //for testing purposes
        return;


}
void addfood()
{
    //LOOP WILL RUN UNTIL X2, Y2 ARE SET TO VALUES WHICH ARE NOT ON THE SNAKE//
    do
    {
        x2 = random(MAX_X - MIN_X + 1) + MIN_X;
        y2 = random(MAX_Y - MIN_Y + 1) + MIN_Y;
    }while(x2==x && y2==y || (pos_key[x2][y2]=='w' || pos_key[x2][y2]=='s' ||
                 pos_key[x2][y2]=='a' || pos_key[x2][y2]=='d'   ));
    gotoxy(x2, y2);
    textcolor(LIGHTMAGENTA);
    cprintf("@");
    textcolor(WHITE);
    return;

}
int checkfood()
{
    if(x==x2 && y==y2)
    {
        return 1;
    }
    else
    {
        return 0;
    }
}
int checkdie()
{
////IF POS_KEY[X][Y] IS ALREADY MAPPED WITH A////
////DIRECTION VALUE WHEN THE HEAD REACHES (X, Y) COORDINATES////
    if(pos_key[x][y]=='w' || pos_key[x][y]=='s' ||
       pos_key[x][y]=='a' || pos_key[x][y]=='d'   )
    {
        return 1;
    }
    else
    {
        return 0;
    }
}
void printgameover()
{
    if(checkdie()!=0)
        {
            int k = 1;
            delay(frame_width);
            //LOOP TO FLASH "GAME OVER" 4 TIMES//
            do
            {
                textcolor(RED);
                gotoxy((MAX_X - MIN_X + 1)/2 - 5 , (MAX_Y - MIN_Y + 1)/2);
                cprintf("GAME OVER!!\a");
                textcolor(WHITE);
                delay(400);
                clrscr();   frame();
                delay(600);
                ++k;
            }while(k<=4);
            while(kbhit())      //to ignore keystrokes pressed while displaying game over//
            {                   //so that final score can be visible//
                getch();
            }
            textcolor(RED);
            gotoxy((MAX_X - MIN_X + 1)/2 - 5 , (MAX_Y - MIN_Y + 1)/2);
            cprintf("GAME OVER!!");
            textcolor(WHITE);
            gotoxy((MAX_X - MIN_X + 1)/2 - 8 , (MAX_Y - MIN_Y + 1)/2 + 1);
            cout<<"Final Score: "<<snaklen*10;
            gotoxy((MAX_X - MIN_X + 1)/2 - 12 , (MAX_Y - MIN_Y + 1)/2 + 2);
            cout<<"Press any key to exit...";
            getch();
            ++counter;
            main();
        }
}
char getdir()
{
    char ch = getch();
    if(ch==0)
    {
        ch = getch();
        switch(ch)
        {
            case 'H':
                return 'w';
```

```c
            case 'P':
                return 's';
            case 'K':
                return 'a';
            case 'M':
                return 'd';
            default :
                return 'x';
        }
    }
    else if(ch==27)
    {
        return ch;
    }
    else
    {
        return 'x';     //any random character, so that it can be ignored by keystroke()
    }
}
void screen()
{
    struct text_info info;
    gettextinfo(&info);

    MAX_X = (int) info.winright - 1;
    MIN_X = (int) info.winleft + 1;
    MAX_Y = (int) info.winbottom - 2;
    MIN_Y = (int) info.wintop + 1;
    max_xs = (int) MAX_X * 0.6;
    max_xm = (int) MAX_X * 0.8;
    max_xl = (int) MAX_X;
    max_ys = (int) MAX_Y * 0.6;
    max_ym = (int) MAX_Y * 0.8;
    max_yl = (int) MAX_Y;
    frame();
    return;
}
void frame()
{

    int width = MAX_X - MIN_X + 3;
    int height = MAX_Y - MIN_Y + 4;
    textcolor(YELLOW);
    gotoxy(1,1);
    for(int i = 0; i < width; i++)
    {
        cprintf("%c", '*');
    }

    for(i = 2; i <= height - 2; i++)
    {
        gotoxy(1, i); cprintf ("%c", '*');
        gotoxy(width, i); cprintf ("%c", '*');
    }

    gotoxy(1, height - 1);
    for(i = 0; i < width; i++)
    {
        cprintf ("%c", '*');
    }
    textcolor(WHITE);
}
int pause()
{
    textcolor(GREEN);
    gotoxy((MAX_X-MIN_X+1)/2 - 15, MIN_Y - 1);
    cprintf("PAUSED*Press esc again to exit");
    gotoxy((MAX_X-MIN_X+1)/2 - 15, MAX_Y + 1);
    cprintf("Press any other key to resume");
    textcolor(WHITE);
    char ch = getch();
    frame();
    if(ch==27)
    {
        return 1;
    }
    else
    {
        return 0;
    }
}
```

```cpp
int select(int x_opt, int y_opt, int pos_gap[], int totalgap)
//X_OPT - X COORDINATE OF THE BULLET IN ALL POSITIONS//
//Y_OPT - Y COORDINATE OF BULLET FOR FIRST OPTION//
//POS_GAP[Y] - THE GAP B/W THE OPTION AT (Y+1) COORDINATE AND THE NEXT OPTION TO IT//
//TOTALGAP - THE TOTAL LINES OCCUPIED BY ALL OPTIONS//
{
    char ch = 26;
    int y_init = y_opt;
    int x_init = x_opt;
    gotoxy(x_init, y_opt);
    cprintf("%c", ch);
    do
    {
        //gotoxy(2, 2);cout<<x_init<<" "<<y_init<<" "
        //cout<<x_opt<<" "<<y_opt<<" "<<pos_gap[y_opt-y_init];  //for testing purposes
        char c = getch();
        if(c==0)
        {
            c = getch();
            gotoxy(x_init, y_opt);
            cout<<' ';          //to delete initial bullet
            //TO SET Y COORDINATE OF NEW BULLET ACCORDING TO ARROW KEY PRESSED//
            switch(c)
            {
                case 'H' :
                    if(y_opt==y_init)
                    {
                        y_opt = y_init + totalgap - 1;
                    }
                    else
                    {
                        y_opt = y_opt - pos_gap[y_opt-y_init - 1];
                    }
                    break;
                case 'P' :
                    if(y_opt==(y_init + totalgap - 1))
                    {
                        y_opt = y_init;
                    }
                    else
                    {
                        y_opt = y_opt + pos_gap[y_opt-y_init];
                    }
                    break;
            }
            gotoxy(x_init, y_opt);      //to print new bullet
            cprintf("%c", ch);
        }
        else if(c==13)
        {
            return y_opt - y_init + 1;  //to return option value according
                                //to y coordinate of bullet
        }
    }while(1);
}
```